(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 762 385 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
12.03.1997 Bulletin 1997/11

(51) Int Cl.$^6$: **G10L 5/06**

(21) Application number: **96306255.9**

(22) Date of filing: **29.08.1996**

(84) Designated Contracting States:
**DE FR GB IT**

(30) Priority: **30.08.1995 US 521543**
**13.11.1995 US 559190**

(71) Applicant: **Dragon Systems Inc.**
**Newton, Massachusetts 02160 (US)**

(72) Inventors:
- **Gadbois, Gregory J.**
  **Newton, Massachusetts 02160 (US)**
- **Van Even, Stijn A.**
  **Newton, Massachusetts 02160 (US)**

(74) Representative: **Deans, Michael John Percy**
**Lloyd Wise, Tregear & Co.,**
**Commonwealth House,**
**1-19 New Oxford Street**
**London WC1A 1LW (GB)**

## (54) Speech recognition

(57) A method of speech recognition includes recognizing a first utterance, recognizing a second utterance having information that is related to the first utterance, and determining the most probable first and second utterances based on stored information about valid relationships between possible first and second utterances. The recognized first utterance may be recognized continuously and the recognized second utterance may be recognized discretely. The determination of the most probable utterances may include creating a list of possible utterances that could be confused with a recognized utterance and rerecognition of a list of possible utterances against an utterance

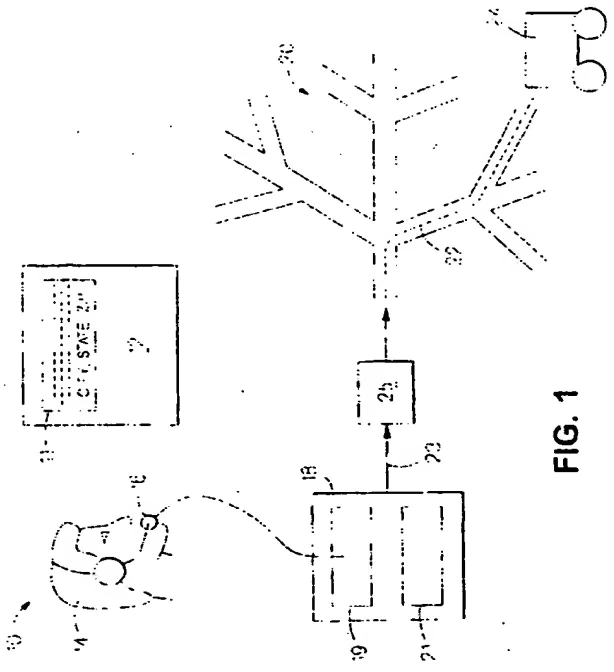


FIG. 1

Best Available Copy

EP 0 762 385 A2

## Description

The invention relates to speech recognition.

A speech recognition system attempts to determine. either on a continuous or discrete basis. what words were intended. based on analysis of a speaker's utterances. A variety of techniques have been used to improve the accuracy of the recognition.

In one aspect, the invention features a method of speech recognition including recognizing a first utterance. recognizing a second utterance having information that is related to the first utterance. and determining the most probable first and second utterances based on stored information about valid relationships between possible first and second utterances.

The method may further include determining the validity of a recognized utterance and including an invalid utterance in a list of possible utterances for comparison with the possible utterances in the list.

In some embodiments of the invention, the determination of the most probable utterances may include rerecognition of a list of possible utterances against at least one utterance.

Implementations of the method may further include the following features: ranking the list of possible utterances based upon how closely each possible utterance corresponds to the at least one utterance: creating an hypothesized list of possible utterances that relate to the at least one recognized utterance based on the stored information: comparing the ranked list of possible utterances to the hypothesized list of possible utterances for commonality between the lists: creating an hypothesized list of possible utterances that relate to the at least one recognized utterance based on the stored information: creating a list of possible utterances that could be confused with the at least one recognized utterance.

In some embodiments of the invention. the recognized first utterance is recognized continuously and the recognized second utterance is recognized discretely.

Implementations of this aspect of the method may further include the following features. A list is created of possible first utterances that may be confused with the recognized first utterance. An hypothesized list of possible second utterances is created that relate to the possible first utterances based on the stored information. The recognized second utterance is added to the hypothesized list of possible second utterances to create a merged list of possible second utterances. The merged list of possible second utterances is rerecognized against the second utterance to get a ranked list of possible second utterances, the ranking based upon how closely each possible second utterance in the merged list corresponds to the second utterance. The ranked list of possible second utterances is compared to the hypothesized list of possible second utterances for commonality between the lists, the highest ranked possible second utterance in the ranked list being compared first. An hypothesized list of possible first utterances is created from the second recognized utterance based on the stored information. The recognized first utterance is added to the hypothesized list of possible first utterances to create a merged list of possible first utterances. The merged list of possible first utterances is rerecognized against the first utterance to get a ranked list of possible first utterances having a ranking based upon how closely each possible first utterance in the merged list corresponds to the first utterance. A possible first utterance ranked second in the ranked list of possible first utterances is evaluated by determining whether a distance parameter associated with the second ranked possible first utterance is within an acceptable limit. A user is advised when the second ranked possible first utterance is not within the acceptable limit and no commonality exists between the ranked list of possible second utterances and the hypothesized list of possible second utterances.

In some embodiments of the invention, the first utterance is a zipstate and the second utterance is a city from a destination address on a package, the determination of the most probable first and second utterances resulting in the sorting of the package according to the packages destination address.

In some embodiments of the invention, the first utterance is a spelled prefix including ordered symbols and the second utterance is a word. A list is created of possible prefixes that could be confused with the recognized prefix. Creating the list of possible prefixes includes determining, in the context of a preceding symbol or silence. a probability of confusing each recognized symbol in the prefix with each symbol in a list of possible symbols. of confusing each recognized symbol in the prefix with an addition of an extra symbol preceding the recognized symbol. and of confusing each recognized symbol in the prefix with the absence of a symbol. Creating the list of possible prefixes includes replacing a sequence of symbols with a single symbol.

In some embodiments of the invention, the first utterance is a spelled word and the second utterance is a word. the determination of the most probable first and second utterances resulting in recognizing the spelled word.

In another aspect. the invention features a method of generating a choice list from a continuously recognized utterance including continuously recognizing a spoken utterance, consulting stored information to determine the probability of confusing possible utterances in the stored information with the recognized utterance, and producing a list of possible utterances from the stored information that could be confused with the recognized utterance.

The method may include rerecognizing the utterance against a merged list of the list of possible utterances and the recognized utterance to create a ranked list of possible utterances having a ranking based upon how closely each utterance in the merged list corresponds to the spoken utterance.

In another aspect, the invention features a method of recognizing ambiguous inputs including recognizing a first ambiguous input. recognizing a second ambiguous input having information that is related to the first ambiguous input, and determining the most probable first and second ambiguous inputs based on stored information about valid relationships between possible first and second ambiguous inputs.

In another aspect. the invention features a method of training a speech recognizer including prompting a user to make a first utterance including symbols, recognizing the symbols, and calculating the probability of confusing each recognized symbol with the prompted symbol. The probabilities are calculated within the context of the preceding symbol or silence.

In another aspect. the invention features a method of displaying word choices during speech recognition including recognizing an uttered word. recognizing a spelling of a prefix of the word, whereby symbols are used to spell the prefix. and displaying a list of word choices on a screen for selection, a top choice on the list corresponding to a highest ranked choice. The symbols are letters, digits, and punctuation.

Advantages of the invention may include one or more of the following. Recognition accuracy is improved considerably over prior speech recognition systems. Any two utterances containing related data can be recognized with improved accuracy. The invention can hypothesize a list of choices from a continuously recognized utterance. The invention can determine if an utterance contains invalid data. The invention can improve the ability of a recognizer to determine a word that has been spelled even though the individual letters ("b" versus "e") may be difficult to distinguish.

Other advantages and features will become apparent from the following description and from the claims.

## Description

Figure 1 is a schematic diagram of a package sorting system.

Figure 2 is a functional block diagram of a city, state and zip code determining system.

Figure 3 is a functional block diagram of a confusability algorithm.

Figures 4-4b are flow diagrams showing an implementation of the address determining system.

Figure 5 is a confusability matrix of probabilities.

Figure 6 is an example of calculated confusion values.

Figure 7 is a listing of zip codes and their related confusion values.

Figure 8 is a functional block diagram of a system for determining a word from the spelling of the word.

Figure 9 is a functional block diagram of a recognition system.

Figure 10 is a functional block diagram of a system for determining a word from the spelling of the word. Figure 11 is a functional block diagram of a confusability algorithm.

Figures 12-12b are flow diagrams showing an implementation of the word determining system.

Figures 13-13b are confusability matrices of probabilities.

Figure 14 is an example of calculated confusion values.

Figure 15 is a listing of words and their related confusion values.

Figures 16-16a are forward and backward diagrams representing Baum Welsh training.

Figure 17 is a diagrammatic representation of a user interface.

As seen in Fig. 1, in system 10 for sorting parcels 12, user 14 reads the address 11 on each parcel 12 and a microphone 16 conveys the spoken information to a computer 18. A speech recognizer 19, such as Voice Tools™ or Dragon Dictate™, available from Dragon Systems. Inc., is supplemented with an associated hypothesis algorithm 21 that determines the spoken address based on the output of the recognizer and notifies the user whether the address is valid. To find that an address is valid. in most instances system 10 must be able to determine the existence of a valid city, state and zip code combination (e.g., Somerville, MA 02143) from the speech recognized address: for states with one shipping destination for all cities only a valid state and zip code is required (e.g., Montana 59100).

For a valid address, the zip code information 23 is sent to a second computer 25 which routes the package along a "tree" of sorting conveyors 20, e.g., by tracking the package using electric eyes or machine readable labels. Package 12 is transported along a predetermined path 22 on conveyors 20 leading to a truck 24 going to the package's destination.

For an invalid address. the user may repeat the address or reject the package as having an invalid address.

Referring to Fig. 2. the first step in the sorting process is that the microphone receives the utterances of the "zipstate" 30 and "city" 32 on the package. The zipstate is the zip code and state spoken as one word, e.g., "02143Massachusetts". Using continuous speech recognition, zipstate recognizer 34 determines the zipstate 38 it believes corresponds to the spoken zip code and state. Each number of the zip code is recognized against a vocabulary of numbers zero through nine and the state is recognized against a vocabulary of states. If the recognizer is not able to recognize five digits and a state, the address is rejected as invalid and an invalid address signal 35 is sent to a sound generator 36 in the headphones which produces a beep 37 signaling the user that the address is invalid.

If the recognizer is able to recognize five digits and a state. a confusability algorithm 39 (described below) is run to create a list of all hypothesized cities 40 that correspond to the recognized zipstate 38 and to other zipstates that the recognizer would be likely to confuse with the recognized zipstate (these other zipstates are limited to valid zip codes within the recognized state). To this list of hypothesized cities 40 is added a list of cities 42 determined by a discrete city recognizer 41. The uttered city 32 is recognized against a vocabulary of valid cities 43 that correspond to the recognized state to produce the list 42 of possible cities.

The list of hypothesized cities 40 and the list of cities 42 are merged 44 to create a city list 45. City list 45 is then used as a city vocabulary against which the city utterance is rerecognized 46 and a ranked list of cities 47 is generated based on a determination by the recognizer of how closely it believes each city in the merged city list 45 matches the uttered city 32. The top city in the ranked list of cities 47, i.e.. the best recognized city, is then compared 48 to the hypothesized cities 40. If the top city is in the hypothesized city list, that city is deemed the correct destination, and the package is routed to that city. If not, the next city in the ranked list is compared to the hypothesized city, and so on: the first match determines the destination city 49 which is delivered to the second computer 25.

If no commonality exits between the ranked list of cities 47 and the hypothesized cities 40, zipstates 51 are hypothesized 50 from the ranked list of cities 47 by consulting a database 52 of zip codes and cities. The originally recognized zipstate 38 is merged 53 with the hypothesized zipstates 51 and the resulting zipstate list 54 is used as a zipstate vocabulary against which the zipstate utterance 30 is discretely rerecognized 55 to generate a ranked list of zipstates 56 including a distance parameter associated with each zipstate signifying how poorly each zipstate was recognized as compared to the highest ranked zipstate. The highest ranked zipstate will. naturally. be the originally recognized zipstate 38. Since it is already known that no hypothesized city from the original zipstate 38 corresponds to a recognized city (if it had. the city would have be in both the ranked list of cities 47 and the hypothesized cities 40 and this step in the algorithm would not have been reached), it is the second zipstate on the ranked list that is evaluated 57. The top ranked zipstate is maintained in ranked list 56 for comparison with the other zipstates in the list to determine the distance parameter. If the distance parameter for the second zipstate is beyond an acceptable limit, e.g.. if $8\log_2$confusion value (described below) of the second zipstate is more than 100 off that of the first zipstate. the package is rejected as having an invalid address and an invalid address signal is sent to the user. If the distance parameter is within the acceptable limit. the package is deemed to have a valid address and the information if delivered to second computer 25 for controlling shipment.

As seen in Fig. 3. in the confusability algorithm 39, the first step is for a zip code hypothesizer 58 to hypothesize possible valid zip codes 59 for the recognized state of zipstate 38 by consulting a database 60 of zip codes and states. The probability that the recognizer would confuse the recognized zip code with each of the hypothesized valid zip codes 59 is determined 61. The top forty zip codes. i.e.. those with the highest probabilities of being confused with the recognized zip code, form a list of zipstates 62 that are then rerecognized 63 against the uttered zipstate 30 in a discrete recognition creating a ranked list 64 of the top ten zipstates. It is from this ranked list 64 of zipstates that the list of hypothesized cities 40 is created by a city hypothesizer 65 which consults a database 66 of zip codes and corresponding cities.

The discrete recognition against the list of zipstates 62 has a higher recognition accuracy rate than the original continuous recognition of the uttered zipstate. Discrete recognition of the original uttered zipstate is not done because it would require recognition against a vocabulary of 5X10^6 zipstates.

An overview of the code is shown in Figs. 4-4b. The user utters the zipstate and city 70. The zipstate is continuously recognized and if the recognized zipstate 38 does not contain good data, i.e., five digits and a state, it is rejected as a bad address 72 and the user is informed that the address was rejected.

If the recognized zipstate contains good data 74. the confusability algorithm 39 is employed to create zipstate list 62. Referring to Figs. 4a and 4b, a propagation subroutine 78 is entered to hypothesize the zip codes which are to populate zipstate list 62.

As seen in Fig. 4b, in the propagation routine 78, a variable n corresponding to the digit place in the zip code (first, second. third, fourth, or fifth number of the zip code) is incremented 81 each time the propagate routine is entered at arrow 80. An index number i corresponding to the numbers zero through nine is incremented 82 after each loop 83 of the propagation routine. At the start of each loop 84, the index number is put into the nth digit place. The built code is checked for validity 85 against the database of valid zip codes for the recognized state. If the index number in the nth digit place is not valid. the index number is incremented and the next number is tried. If the index number in the nth digit place is valid, a confusion value 88 corresponding to how likely it would be for the recognizer to confuse the number it recognized in the nth digit place with the index number in that digit place is calculated (described further below). The code is placed in a cue 90 and the loop is repeated incrementing the index number each time until i=9 has been checked 91 for the nth digit place. The codes are placed in queue 90 in descending order of probability.

When i=9, the propagation routine is exited. In Fig. 4a, if there are less than forty zip codes 93 in a top list 92, the highest probability code 94 in queue 90 is considered. If code 94 has less than five digits 95. the propagation routine is reentered with this code. Variable n is incremented and the process is repeated until there are 5 digits in the highest probability code. This zip code is then moved to the top list 92. The next highest probability code 94 is then taken from queue 90 for consideration. If there are five digits in this zip code, the zip code is moved to the top list. This is repeated until the code at the top of queue 90 has less than five digits. This partial zip code is then propagated. The confusability algorithm is repeated until there are forty zip codes in top list 92.

Referring again to Fig. 4. the list of zipstates 62 corresponds to the forty zip codes from top list 92 combined with the recognized state. The zipstates 62 are then rerecognized by a discrete recognition 95' against the original utterance. The recognizer returns a distance parameter of how well the zipstates were recognized 96. Beyond a limit the zipstates are rejected. At most the top ten recognized zipstates form the ranked list of zipstates 64. A database of cities corresponding to zip codes is then consulted to create the list of hypothesized cities 40.

At this point in the code (though the following check could be done at any time after recognizing the original zipstate utterance) the recognized zip code is checked against the recognized state. If the recognized zip code is a valid zip code for the recognized state 97, and if there is only one shipping destination for the recognized state 98. the package is shipped to the recognized state. If either the recognized zip code is not valid for the recognized state or there is more than one shipping destination for the recognized state. the city is recognized 41 by discrete recognition forming the list of cities 42. From the list of cities 42 and the hypothesized cities 40, the ranked list of cities 47 is created as described above with reference to Fig. 2. The algorithm proceeds as described above.

Referring to Fig. 5, the confusion value corresponding to how likely it would be for the recognizer to confuse the recognized number in the nth digit place with the index number in the nth digit place is calculated from a confusability matrix of probabilities. The rows of the matrix correspond to the recognized number and the columns of the matrix correspond to the index number. For example, the probability of confusing the recognized number 4 with the index number 2 is 0.0216. As the number of digit places increases. the probability is calculated by multiplying the probabilities for each digit place. For example. the probability of confusing the recognized number 44 with 22 is 0.0216 x 0.0216 = .00047: the probability of confusing 442 with 223 is .00047 x .1888.

The confusability matrix is experimentally determined by having predetermined numbers read off to the speech recognizer and calculating how often the number is incorrectly recognized and what number it is incorrectly recognized as. The confusability matrix may be customized for an individual user. The confusability matrix is shown in Fig. 5 as a matrix of probabilities for ease of description: the code utilizes a matrix of natural logs of the probability.

EXAMPLE

Utterance: 02143Massachusetts Somerville
Zipstate recognized: 03112MA
Confusability algorithm:

Referring to Fig. 6 in which the calculation of the confusion values is shown for each digit place. in the first digit place, n=1, 0 is the only valid number for Massachusetts zip codes. Queue 90 becomes:

| n | i | zip code | confusion value |
|---|---|---|---|
| 1 | 0 | 0---- | 1 |

There are fewer than 40 zip codes in top list 92: 0---- is pulled from queue 90 as the highest probability code in the queue: there are less than five digits in the pulled code: the propagation routine is reentered.

In the second digit place, n=2, 1 and 2 are the only valid digits. Queue 90 becomes:

| n | i | zip code | confusion value |
|---|---|---|---|
| 2 | 2 | 02--- | 0.0497 |
| 2 | 1 | 01--- | 0.0001 |

02--- is pulled from queue 90 as the highest probability code and the propagation routine is reentered.

In the third digit place, n=3, 0 to 7 are valid digits. Queue 90 becomes:

| n | i | zip code | confusion value |
|---|---|---|---|
| 3 | 1 | 021-- | 0.04970 |
| 3 | 4 | 024-- | 0.00456 |
| 3 | 5 | 025-- | 0.00365 |

(continued)

| n | i | zip code | confusion value |
|---|---|---|---|
| 3 | 7 | 027-- | 0.00209 |
| 3 | 0 | 020-- | 0.00077 |
| 2 | 1 | 01--- | 0.00010 |

For i= 2. 3 and 6 the confusion value is 0.0497 * 0.0001. which is so low that these codes get dropped from the queue.

021-- is pulled from queue 90 as the highest probability code and the propagation routine is reentered.

In the forth digit place, n=4, 0 to 9 are valid digits. Queue 90 becomes:

| n | i | zip code | confusion value |
|---|---|---|---|
| 4 | 1 | 0211- | 0.04970 |
| 4 | 4 | 0214- | 0.00456 |
| 3 | 4 | 024-- | 0.00456 |
| 4 | 5 | 0215- | 0.00365 |
| 3 | 5 | 025-- | 0.00365 |
| 4 | 7 | 0217- | 0.00209 |
| 3 | 7 | 027-- | 0.00209 |
| 4 | 9 | 0219- | 0.00150 |
| 4 | 0 | 0210- | 0.00077 |
| 3 | 0 | 020-- | 0.00077 |
| 2 | 1 | 01--- | 0.00010 |

0211- is pulled from queue 90 as the highest probability code and the propagation routine is reentered.

In the fifth digit place, n=5, 0 to 9 are valid digits. Queue 90 becomes:

| n | i | zip code | confusion value |
|---|---|---|---|
| 5 | 2 | 02112 | 0.04970 |
| 5 | 3 | 02113 | 0.00940 |
| 5 | 6 | 02116 | 0.00600 |
| 4 | 4 | 0214- | 0.00456 |
| 3 | 4 | 024-- | 0.00456 |
| 5 | 0 | 02110 | 0.00365 |
| 4 | 5 | 0215- | 0.00365 |
| 3 | 5 | 025-- | 0.00365 |
| 5 | 7 | 02117 | 0.00260 |
| 5 | 8 | 02118 | 0.00209 |
| 4 | 7 | 0217- | 0.00209 |
| 3 | 7 | 027-- | 0.00209 |
| 5 | 4 | 02114 | 0.00200 |
| 5 | 9 | 02119 | 0.00150 |
| 4 | 9 | 0219- | 0.00150 |
| 5 | 5 | 02115 | 0.00140 |
| 5 | 1 | 02111 | 0.00100 |
| 4 | 0 | 0210- | 0.00077 |
| 3 | 0 | 020-- | 0.00077 |
| 2 | 1 | 01--- | 0.00010 |

There are fewer than 40 zip codes in top list 92: 02112 is pulled from queue 90 as the highest probability code: 02112 has five digits so it is moved to top list 92: 02113 is pulled from queue 90 and moved to the top list: 02116 is pulled from queue 90 and moved to the top list: 0214- is pulled from queue 90 and propagated.

Again, in the fifth digit place, n=5. 0 to 9 are valid digits. Queue 90 becomes:

| n | i | zip code | confusion value |
|---|---|---|---|
| 5 | 2 | 02142 | 0.00456 |
| 3 | 4 | 024-- | 0.00456 |
| 5 | 0 | 02110 | 0.00365 |
| 4 | 5 | 0215- | 0.00365 |
| 3 | 5 | 025-- | 0.00365 |
| 5 | 7 | 02117 | 0.00260 |
| 5 | 8 | 02118 | 0.00209 |
| 4 | 7 | 0217- | 0.00209 |
| 3 | 7 | 027-- | 0.00209 |
| 5 | 4 | 02114 | 0.00200 |
| 5 | 9 | 02119 | 0.00150 |
| 4 | 9 | 0219 | 0.00150 |
| 5 | 5 | 02115 | 0.00140 |
| 5 | 1 | 02111 | 0.00100 |
| 5 | 3 | 02143 | 0.00086 |
| 4 | 0 | 0210- | 0.00077 |
| 3 | 0 | 020-- | 0.00077 |
| 5 | 6 | 02146 | 0.00055 |
| 5 | 0 | 02140 | 0.00033 |
| 5 | 7 | 02147 | 0.00020 |
| 5 | 8 | 02148 | 0.00019 |
| 5 | 4 | 02144 | 0.00018 |
| 5 | 9 | 02149 | 0.00014 |
| 5 | 5 | 02145 | 0.00013 |
| 2 | 1 | 01--- | 0.00010 |

02142 is moved to the top list and 024-- is pulled from queue 90 and propagated. The process is continued until there are forty zip codes in top list 92. The top 30 zip codes are shown in Fig. 7.

Discrete recognition of the top forty zipstates yields four zipstates in ranked order, corresponding cities 40 are hypothesized:

| zipstates | hypothesized cities 40 |
|---|---|
| 02143MA | Somerville |
| 02147MA | Brookline Village |
| 02112MA | Essex Station |
| 02113MA | Hanover |

Cities 42 are discretely recognized from the utterance:

Sagamore
Somerville
Sudbury
Salisbury
Sheldonville

City list 44 is formed:

Somerville
Brookline Village
Essex Station
Hanover
Sudbury
Sagamore
Salisbury
Sheldonville

Ranked city list 47 is formed from discrete rerecognition of the city:

Sagamore
Somerville
Sudbury
Salisbury
Sheldonville

Comparison of hypothesized cities 40 to ranked list 47 shows Somerville as the common city. The package is shipped to Somerville.

Factory tests of the invention for sorting packages have resulted in a package rejection rate for invalid address of about 2% and a misshipping rate of about 0.1%: whereas, factory tests of the invention without employing the confusability algorithm, i.e., hypothesizing cities only from the one recognized zipstate, resulted in a package rejection rate of about 5-8% and a misshipping rate of about 0.1%.

Referring to Fig. 8. a similar hypothesis algorithm can be used to recognize spoken letters when spelling a word. The word 100 is first uttered and recognized 101 by discrete recognition. If the recognizer gets the word wrong. the user so indicates to the recognizer and then utters the spelling of the word 102. The spoken string of letters is recognized 104 by continuous recognition. The confusability algorithm 106 uses the probability that a recognized letter 105 would be confused with a letter of the alphabet. As in the zip code example, a ranked list of letter combinations 107 is formed (similar to ranked zipstate list 64). In creating the ranked list of letter combinations. a dictionary database 108 is consulted to validate that each letter combination is capable of forming a valid word (similar to checking for valid zip codes). The top forty valid letter combinations are then discretely rerecognized against the spoken string of letters to form ranked list 107. A list of words 110 is then hypothesized from the ranked list of letter combinations (similar to hypothesized cities 40).

The hypothesized words 110 and a list of words 111 formed by discrete recognition of the uttered word 100 are merged 112 to form a word list 113. Word list 113 is discretely rerecognized 114 against the spoken word and a ranked list of words 115 is formed. Ranked list of words 115 is compared 116 to hypothesized words 110, the top ranked word being compared first. The first ranked word that shows up in the hypothesized words 110 is chosen 117.

If the chosen word is incorrect the user can reject it and the next ranked word that shows up in the hypothesized words 110 is chosen. This process can be continued until the correct word is chosen or the ranked list is exhausted. Alternatively, word comparer 116 can create a choice list 117' containing all words in common between ranked list of words 115 and hypothesized words 110 can be displayed to the user. If the correct word is contained within the choice list, the user can select the word either using the keyboard or with a spoken command. For example, if the misrecognized uttered word 100 is "in", a choice list may appear as:

in:

in
an
ink
imp
and
amp

with the first ranked word appearing above the list and within the list. If the user then continues speaking without choosing from the choice list, the first ranked word is automatically chosen.

If the ranked list is exhausted without finding a common word or the correct word, possible spellings 119 are hypothesized 118 from the ranked list of words 115. The recognized spelling 104 is merged 120 with the hypothesized spellings 119 and the resulting spelling list 121 is discretely rerecognized 122 against the original spelling utterance 102 to form a ranked list of spellings 123. The second spelling on the ranked list is evaluated 124. If the distance parameter returned by the recognizer is beyond the acceptable limit, the spelling is rejected. if it is within the acceptable limit. that word is chosen. If the user rejects the spelling, the next spelling on the list can be considered. Alternatively, as described above. a choice list 123' containing all but the first ranked spelling in the ranked list of spellings 123 can be displayed to the user for selection.

Referring to Fig. 10. a particular embodiment of a hypothesis algorithm for recognizing spelled word prefixes is shown. Symbols which may be included in the prefix are generally letters. numbers. and punctuation (e.g.. hyphen. period. comma. brackets). The word 232 is first uttered and recognized 241 by discrete recognition. If word 232 is incorrectly recognized. the user utters the spelling of the prefix of the word 230 (generally 3 to 5 symbols. the prefix may be part or all of the word). The spelled prefix is recognized 234 by continuous recognition. The confusability algorithm 239 uses the probability that the symbols in recognized prefix 238 would be confused with other symbols to hypothesize a ranked list of prefixes 264. A distance parameter associated with each prefix signifies how poorly each prefix was recognized as compared to the highest ranked prefix. A top ranked prefix 242 determined by discrete recognition of the uttered word 232 is merged 244 with prefix list 264 to form merged prefix list 245. Word hypothesizer 265 hypothesizes a word list 240 by consulting a database of prefixes and corresponding words 266.

Word list 240 is discretely rerecognized 246 against the uttered word 232 to form a ranked list of words 247 including a distance parameter associated with each word signifying how poorly each word was recognized as compared to the highest ranked word. The distance parameters associated with prefixes 264 are then added 248 to the distance parameters of words 247 that include corresponding prefixes and ranked list 247 is reranked 249 according to the new distance parameters. This aids in the recognition of homophones because it gives a higher ranking to the homophone having a higher ranked spelled prefix. Reranked word list 249 is displayed to the user as a choice list. If the correct word is contained within the choice list. the user can select the word either using the keyboard or with a spoken command.

As seen in Fig. 11. in the confusability algorithm 239. a prefix hypothesizer 258 hypothesizes a list of prefixes 262 based on the probability that recognizer 234 would confuse recognized prefix 238 with other word prefixes. A database 260 of valid prefixes or words is used as a filter to form a list of valid prefixes 262. Database 260 can be. e.g.. a list of valid prefixes. a dictionary. or an n-letter language model. The top one-hundred and fifty prefixes. i.e.. those with the highest probabilities of being confused with the recognized prefix 238. form the list of prefixes 262. The list of prefixes 262 are then rerecognized 263 against the uttered prefix 230 in a discrete recognition which filters out poorly confused prefixes and creates a ranked list 264 of the top fifteen to twenty prefixes. It is this ranked list 264 of prefixes that is merged with the top ranked prefix from the discrete recognition of the uttered word to form the merged prefix list 245 from which the list of hypothesized words 240 is created.

An overview of the code is shown in Figs. 12-12b. The user utters a word and if the recognizer misrecognizes the word. the user then utters the prefix 270. The confusability algorithm 239 is then employed to create prefix list 262. Referring to Figs. 12a and 12b. a propagation subroutine 278 is entered to hypothesize the prefixes which are to populate prefix list 262.

As seen in Fig. 12b. in the propagation routine 278. a variable n corresponding to the digit place in the prefix (first. second. third. forth. or fifth place in a prefix having five symbols) is incremented 281 each time the propagate routine is entered at arrow 280 (unless the prior propagation was an insertion 380, as described below, in which case n is not incremented 381). Alternatively, variable n can be incrementing at the end of each permutation loop and deletion and kept constant at the end of each insertion loop. Propagation routine 278 includes three sub-routines. a permutation routine 378. an insertion routine 478 and a deletion routine 578. An index value i (i = 0 to the number of symbols. i= 0 to 35 in the example that follows) corresponding to the symbols letters a through z. digits zero through nine. and punctuation. in that order. is incremented 382. 482 after each loop 383. 483 of the sub-routines. As described more fully below. confusability is determined in the context of the preceding symbol in the prefix.

At the start 384 of permutation routine 378. the index value i is put into the nth digit place and checked for validity 385 against the database of valid prefixes 260. If the index value in the nth digit place is not valid. the index value is incremented and the next value is tried. If the index value in the nth digit place is valid. a confusion value 388 corresponding to how likely it would be for the recognizer to confuse the value it recognized in the nth digit place with the index value in that digit place is calculated (described further below). The prefix is placed in a queue 390 and the loop is repeated incrementing the index value each time until i=38 has been checked 391 for the nth digit place. The codes are placed in queue 390 in descending order of probability.

When i=38 in permutation routine 378. if the last loop of propagation routine 278 entered for this build was not a deletion loop 480, insertion routine 478 is entered 484. i is set to zero and put into the nth digit place (insertion loops do not follow deletion loops and deletion loops do not follow insertion loops because to do so would cancel out their effect). The built prefix is checked for block permutation 486 and for validity 485. If the index value in the nth digit place is valid, a confusion value 488 is calculated and the prefix is placed in queue 390. Loop 483 is repeated incrementing the index value each time until i=38 has been checked 491 for the nth digit place.

When i=38 in insertion routine 478, if the last loop of propagation routine 278 entered for this build was not an insertion loop 580. deletion routine 578 is entered 584. A confusion value 588 corresponding to how likely it would be for the recognizer to confuse the value it recognized in the nth digit place with the absence of a symbol in the nth digit place is calculated. The prefix is placed in queue 390.

At the end of the propagation routine. the prefix that the propagation routine is entered with (described more fully below) is checked for inclusion of a block permutation 686. A block permutation occurs if the built prefix includes LU, LE, MU or ME. The letter W is substituted for these letter combinations and checked for validity 685 against the database of valid prefixes 260. If it is a valid prefix it is placed in queue 390.

Propagation routine 278 is then exited. Referring to Fig. 12a. the highest probability prefix 294 in queue 390 is considered. If the value of n from the prior propagation of prefix 294 is less than the number of symbols in recognized prefix 238. the propagation routine is reentered with this prefix. If the last build for this prefix was not an insertion, variable n is incremented and the propagation routine reentered. If the last build for this prefix was an insertion. the propagation routine is reentered without incremented variable n. The process is repeated until the value of n of the highest probability prefix in queue 390 is equal to the number of symbols in recognized prefix 238. This prefix is then moved to the top list 292. The next highest probability prefix 294 is then taken from queue 390 for consideration. The confusability algorithm is repeated until queue 390 is exhausted or there are one-hundred and fifty prefixes 293 in top list 292.

Referring again to Fig. 12. the list of prefixes 262 corresponds to the one-hundred and fifty prefixes from top list 292. The prefixes 262 are then rerecognized by a discrete recognition 295 against the original prefix utterance 230. The recognizer returns a distance parameter with each prefix related to how well the prefixes were recognized. At most the top fifteen to twenty recognized prefixes form the ranked list of prefixes 264. The top prefix 242 from discrete recognition 241 of uttered word 232 is combined with prefix list 264. A database of words corresponding to prefixes is then consulted to create the list of hypothesized words 240. The ranked list of words 247 is created as described above with reference to Fig. 10. The algorithm proceeds as described above.

Referring to Figs. 13-13b, a few examples of confusability matrixes are shown. Figure 13 includes confusion values for permutations. Fig. 13a includes confusion values for insertions. and Fig. 13b includes confusion values for deletions. The confusability matrices are shown as matrices of probabilities for ease of description: the code utilizes matrices of natural logs of the probability. The symbols hyphen, period and comma are not shown.

The confusion value corresponding to how likely it would be for the recognizer to confuse the recognized symbol in the nth digit place with the index value in the nth digit place is determined in the context of the symbol in the preceding nth digit place (silence if n=1). The rows of the matrix correspond to the symbol context, the columns of the matrix correspond to the index value, and the header symbol 510 corresponds to the recognized symbol. There are 39 such matrixes, one for each symbol, for permutation, insertion and deletion for a total of 117 matrixes or 39 three dimensional matrixes.

For example. in the case of a permutation, the probability of confusing the recognized letter B with the index value D in the context of silence is 0.011988 and the probability of confusing the recognized letter B with the index value e in the context of a is 0.017751. The probability of confusing the recognized letter O with the index value 4 in the context of t is 0.000258. As the number of digit places increases. the probability is calculated by multiplying the probabilities for each digit place. For example. the probability of confusing the recognized prefix BO with TA is 0.001323 x 0.001831 = 0.000002422. The confusability matrices are experimentally determined as described below.

EXAMPLE

Uttered word: Dosage
List of recognized word:
   Postage. Post, Postal, Buffalo, Dosing
Spelled prefix: DOS
Prefix recognized: BOS
Confusability algorithm:

Referring to Fig. 14 in which the calculation of the confusion values is shown for each digit place (only those symbols having confusion values greater than 0.01 are shown for simplicity, generally the cut-off value is 0.000010). In the first digit place, n=1, in the context of silence, the valid prefixes are put in queue 390:

| n | operation | prefix | confusion value |
|---|---|---|---|
| 1 | P | B | 0.655987 |
| 1 | D |  | 0.072822 |

(continued)

| n | operation | prefix | confusion value |
|---|---|---|---|
| 1 | P | D | 0.011988 |
| 1 | P | E | 0.011813 |
| 1 | I | E | 0.010596 |

(P = permutation. D = deletion. I = insertion)

$^{P}$B is pulled from queue 390 as the highest probability prefix in the queue: n is less than 3 and the prior operation was not an insertion so n is incremented: the propagation routine is reentered. Queue 390 becomes:

| n | operation | prefix | confusion value |
|---|---|---|---|
| 2 | P | BO | 0.405482 |
| 2 | D | B | 0.095995 |
| 1 | D |  | 0.072822 |
| 2 | I | BB | 0.055001 |
| 2 | I | BO | 0.050121 |
| 2 | P | BB | 0.015694 |
| 1 | P | D | 0.011988 |
| 1 | P | E | 0.011813 |
| 1 | I | E | 0.010596 |
| 2 | P | BW | 0.010545 |

$^{P}$BO is pulled from queue 390 as the highest probability prefix in the queue: n is less than 3 and the prior operation was not an insertion so n is incremented: the propagation routine is reentered. Queue 390 becomes (only those possible prefixes having confusion values greater than 0.002 are shown for clarity):

| n | operation | prefix | confusion value |
|---|---|---|---|
| 3 | P | BOS | 0.218691 |
| 2 | D | B | 0.095995 |
| 1 | D |  | 0.072822 |
| 3 | I | BOO | 0.064946 |
| 3 | D | BO | 0.059754 |
| 2 | I | BB | 0.055001 |
| 2 | I | BO | 0.050121 |
| 3 | I | BOS | 0.017041 |
| 3 | P | BOO | 0.016986 |
| 2 | P | BB | 0.015694 |
| 1 | P | D | 0.011988 |
| 1 | P | E | 0.011813 |
| 1 | I | E | 0.010596 |
| 2 | P | BW | 0.010545 |
| 3 | P | BOA | 0.004753 |

$^{P}$BOS is pulled from queue 390 as the highest probability prefix in the queue: n equals 3 so BOS is moved to top list 292: $^{D}$B is pulled from queue 390 as the highest probability prefix in the queue: n is less than 3 and the prior operation was not an insertion so n is incremented: the propagation routine is reentered. Queue 390 becomes:

| n | operation | prefix | confusion value |
|---|---|---|---|
| 1 | D | | 0.072822 |
| 3 | I | BOO | 0.064946 |
| 3 | D | BO | 0.059754 |
| 2 | I | BB | 0.055001 |
| 2 | I | BO | 0.050121 |
| 3 | I | BOS | 0.017041 |
| 3 | P | BOO | 0.016986 |
| 2 | P | BB | 0.015694 |
| 1 | P | D | 0.011988 |
| 1 | P | E | 0.011813 |
| 1 | I | E | 0.010596 |
| 2 | P | BW | 0.010545 |
| 3 | D | B | 0.009121 |
| 3 | P | BOA | 0.004753 |

$^{D}$ is pulled from queue 390 as the highest probability prefix in the queue: n is less than 3 and the prior operation was not an insertion so n is incremented: the propagation routine is reentered. Queue 390 becomes:

| n | operation | prefix | confusion value |
|---|---|---|---|
| 3 | I | BOO | 0.064946 |
| 2 | P | O | 0.063608 |
| 3 | D | BO | 0.059754 |
| 2 | I | BB | 0.055001 |
| 2 | I | BO | 0.050121 |
| 3 | I | BOS | 0.017041 |
| 3 | P | BOO | 0.016986 |
| 2 | P | BB | 0.015694 |
| 1 | P | D | 0.011988 |
| 1 | P | E | 0.011813 |
| 1 | I | E | 0.010596 |
| 2 | P | BW | 0.010545 |
| 3 | D | B | 0.009121 |
| 2 | D | | 0.006324 |
| 3 | P | BOA | 0.004753 |

$^{I}$BOO is pulled from queue 390 as the highest probability prefix in the queue: n equals 3 so BOO is moved to top list 292: $^{P}$O is pulled from queue 390 as the highest probability prefix in the queue: n is less than 3 and the prior operation was not an insertion so n is incremented: the propagation routine is reentered. Queue 390 becomes:

| n | operation | prefix | confusion value |
|---|---|---|---|
| 3 | D | BO | 0.059754 |
| 2 | I | BB | 0.055001 |
| 2 | I | BO | 0.050121 |
| 3 | P | OS | 0.034306 |
| 3 | I | BOS | 0.017041 |
| 3 | P | BOO | 0.016986 |
| 2 | P | BB | 0.015694 |
| 1 | P | D | 0.011988 |
| 1 | P | E | 0.011813 |
| 1 | I | E | 0.010596 |
| 2 | P | BW | 0.010545 |
| 3 | I | OO | 0.010188 |
| 3 | D | O | 0.009374 |
| 2 | D | B | 0.009121 |
| 2 | D |  | 0.006324 |
| 3 | P | BOA | 0.004753 |
| 3 | I | OS | 0.002673 |
| 3 | P | OO | 0.002665 |

$^{D}$BO is pulled from queue 390 as the highest probability prefix in the queue: n equals 3 so BO is moved to top list 292: $^{I}$BB is pulled from queue 390 as the highest probability prefix in the queue: n is less than 3 and the prior operation was an insertion so n is not incremented: the propagation routine is reentered and does not yield any valid prefixes. $^{I}$BO is pulled from queue 390 as the highest probability prefix in the queue: n is less than 3 and the prior operation was an insertion so n is not incremented: the propagation routine is reentered. Queue 390 becomes:

| n | operation | prefix | confusion value |
|---|---|---|---|
| 3 | P | OS | 0.034306 |
| 2 | P | BOO | 0.028937 |
| 3 | I | BOS | 0.017041 |
| 3 | P | BOO | 0.016986 |
| 2 | P | BB | 0.015694 |
| 1 | P | D | 0.011988 |
| 1 | P | E | 0.011813 |
| 1 | I | E | 0.010596 |
| 2 | P | BW | 0.010545 |
| 3 | I | OO | 0.010188 |
| 3 | D | O | 0.009374 |
| 2 | D | B | 0.009121 |
| 2 | I | BOO | 0.008720 |
| 2 | D |  | 0.006324 |
| 3 | P | BOA | 0.004753 |
| 3 | I | OS | 0.002673 |
| 3 | P | OO | 0.002665 |

$^{P}$OS is pulled from queue 390 and moved to top list 292: $^{P}$BOO is pulled from queue 390 as the highest probability prefix in the queue; n is less than 3 and the prior operation was not an insertion so n is incremented: the propagation routine is reentered. Queue 390 becomes:

| n | operation | prefix | confusion value |
|---|---|---|---|
| 3 | I | BOS | 0.017041 |
| 3 | P | BOO | 0.016986 |
| 2 | P | BB | 0.015694 |
| 3 | P | BOOS | 0.015607 |
| 1 | P | D | 0.011988 |
| 1 | P | E | 0.011813 |
| 1 | I | E | 0.010596 |
| 2 | P | BW | 0.010545 |
| 3 | I | OO | 0.010188 |
| 3 | D | O | 0.009374 |
| 2 | D | B | 0.009121 |
| 2 | I | BOO | 0.008720 |
| 2 | D | [illegible] | 0.006324 |
| 3 | P | BOA | 0.004753 |
| 3 | I | BOOO | 0.004635 |
| 3 | D | BOO | 0.004264 |
| 3 | I | OS | 0.002673 |
| 3 | P | OO | 0.002665 |

$^{I}$BOS is pulled from queue 390 as the highest probability prefix in the queue: n equals 3 but this is a repeat lowered value prefix so it is dropped. $^{P}$BOO is pulled from queue 390 and similarly dropped. Fig. 14 follows the propagation routine through $^{P}$DO. The process is continued until queue 390 is exhausted or there are one hundred and fifty prefixes in top list 292. The top 13 prefixes are shown in Fig. 15.

Discrete rerecognition of the top one hundred and fifty prefixes against the uttered prefix 230 yields three prefixes in ranked order. If the top recognized prefix 242 from the original discrete recognition of the uttered word (POS) is already included in ranked list 264 it is not added. Corresponding words 240 are hypothesized:

| prefixes | hypothesized words 240 |
|---|---|
| BOS | boscage. boskage, bosh, bosk, etc. |
| POS | posada. pose, Poseidon, poser. etc. |
| DOS | dos. dosage, dose. dosido, |
| dosimeter, | doss. dossal, dossier. dost |

Ranked word list 247 is formed from discrete rerecognition against the uttered word 232:

postage
dosage
boscage
boskage
dossal
dossier
poser

The distance parameters associated with prefixes 264 are then added to the distance parameters of words 247 that include corresponding prefixes and ranked list 247 is reranked according to the new distance parameters. The reranked list is then displayed to the user for selection.

Referring to Fig. 17, a user interface 700 permitting a user to select from ranked word list 247 is shown. Here, the word was disco, the recognizer recognized fiscal. the user spelled DIS, and disco appeared as choice 9 in list 247. Selecting 9 on a keyboard (not shown) replaces fiscal with disco.

The confusability matrices are experimentally determined by having a user spell predetermined words to the speech recognizer and comparing the recognized spelling to the known spelling. Referring to Figs. 16-16a, the training algorithm prompts the user to spell BED. The user spells BED and the recognizer recognizes BET. Baum Welsh training in the context of Hidden Markov Models is a standard technique in speech recognition. See. for example. Rabiner. L. R.. and Juang. B.H.. "Introduction to Hidden Markov Models," *IEEE ASSP*, pp. 4-16, January 1986. The invention utilizes a modification of Baum Welsh training to determine the probability of permutations. insertions and deletions for each symbol in the context of each symbol.

Permutations build diagonally, deletions horizontally, and insertions vertically. Permutations feed all three nodal points, deletions feed permutation and deletion nodal points, and insertions feed permutation and insertion nodal points. The points in each nodal point are arranged as (insertion. permutation. deletion). Permutations have an additional factor of 10 if the letter is correct and a factor of 2 if the letter is not correct. In the forward alpha direction of Fig. 16. the silence node starts with values of (1. 1. 1) for (insertion. permutation. deletion) respectively. All other nodal points start with values of (0. 0. 0). In the backward beta direction of Fig. 16a, the lower right hand node starts with values of (1. 1. 1) and all other nodal points start with values of (0. 0. 0). The forward and backward diagrams are used to calculate confusability probabilities as follows:

| node | context | prompt | recog. | alpha | beta | probability |
|---|---|---|---|---|---|---|
| ins | Sil | B | B | 1 | 22 | (1/260*22/260)*10 |
| per | Sil | B | B | 1 | 24 | (1/260*24/260) |
| del | Sil | B | | 1 | 22 | (1/260*22/260) |
| ins | B | E | E | 10 | 2 | (10/260*2/260)*10 |
| per | B | E | E | 10 | 2 | (10/260*2/260) |
| del | B | E | | 10 | 2 | (10/260*2/260) |
| ins | B | E | B | 1 | 2 | (1/260*2/260) |
| per | B | E | B | 1 | 4 | (1/260*4/260) |
| del | B | E | | 0 | 4 | (0/260*4/260) |
| ins | Sil | B | T | 0 | 1 | (0/260*1/260) |
| per | Sil | B | T | 1 | 1 | (1/260*1/260) |
| del | Sil | B | | 1 | 0 | (1/260*0/260) |
| ins | E | D | E | 12 | 0 | (12/260*0/260) |
| per | E | D | E | 12 | 1 | (12/260*1/260) |
| del | E | D | | 2 | 1 | (2/260*1/260) |

Repetitive probabilities for the same node (operation), context, prompt and recognized symbol are added to determine the probability of confusing the recognized symbol with the prompt. This procedure is carried out until the proportions between the confusions becomes more or less stable indicated that a majority of confusion phenomena have been captured. Anomalies in the data can be smoothed by looking at related confusions, i.e., the probability of confusing e with a in the context of b is related to the probability of confusing a with e in the context of b. It is useful to have several speakers participate in the training process to capture a large variety of possible confusion errors. The resulting matrix may be customized for a particular user.

Speech recognition in accordance with the invention can be performed whenever there are two or more utterances that include related data, e.g., knowing the city and state the zip code is known, knowing the spelling of a word the word is known. Other examples of related data are inventory name and number. flight origination/destination and flight number. A database containing the relationships between the data is consulted to hypothesize one set of data from the other. Referring to Fig. 9, a first utterance 150 is recognized 151 by continuous recognition. A confusability algorithm 153 creates a ranked list including entries that the recognizer would be likely to confuse with the recognized utterance 152 and hypothesizes a list of second utterances 155 by consulting a database 154.

A discrete recognizer 157 recognizes a second utterance 156 (second utterance 156 contains data redundant with first utterance 150) by recognizing the utterance against a database 158 and creates a list of possible second utterances 159. The hypothesized list of second utterances 155 and the list of possible second utterances 159 are merged 160 to form a second utterance list 161. Second utterance list 161 is discretely rerecognized 162 against the second utterance 156 and a ranked list of second utterances 163 is formed. Ranked list of second utterances 163 is compared 164 to hypothesized second utterances 155, the top ranked second utterance being compared first. The first ranked second utterance that shows up in the hypothesized second utterances 155 is chosen 165.

If the choice is incorrect or the ranked list is exhausted without finding commonality, possible first utterances are hypothesized 166 from the ranked list of second utterances 163 by consulting a database 167 to form a list of hypothesized first utterances 168. The recognized first utterance 152 is merged 169 with the hypothesized first utterances 168 and the resulting first utterance list 179 is discretely rerecognized 171 against the original first utterance 150 to form a ranked list of first utterances 172. The second first utterance on the ranked list is evaluated 173.

The speech recognizer is able to inform the user of bad data. i.e.. no commonality was found between ranked list of second utterances 163 and hypothesized second utterances 155. and the second first utterance on the ranked list 172 has a distance parameter that is beyond the acceptable limit. Appendix A is the source code in C++. including header files. for the city/state/zip sort application. Appendix B includes additional source code in C++, including header files, for the prefix/word application. As compared to the city/state/zip application. in the prefix/word application. the hypo.h. hashalfa.h and hypo.cpp files have been modified, choice.h has replaced wapp.h and choice.cpp has replaced wapp.cpp. and trie.h and trie.cpp have been added.

Other embodiments are within the scope of the following claims. The first and second utterances can be two parts of one utterance. The recognition system can be used with an optical character recognizer instead of a speech recognizer.

## APPENDIX A

```
////////////////////////////////////////////////////////////////////////////
/*
   Description

        WAPP.CPP
        Main module: Windows loop and algoritm flow in onSpeech()

   Copyright (c) 1991-1995 by Dragon Systems, Inc.

   Author: Greg Gadbois
   Created: 1991 -1995


        FUNCTIONS
        .........

        1. Functions declared in a class and defined in WAPP.CPP:


                                                 member Function
return          Class
Value                                            ...............
......          .....


void            ChoicelistStats          ::recordResult( const char* src )
void            ChoicelistStats          ::printStats()

unsigned        WordIdToCustomCityHS::keyHash( const SLinkBase* slb ) const
int             WordIdToCustomCityHS::compare( const SLinkBase* slb1, const SLin
  => kBase* slb2 ) const


                          FindFilePath          ::FindFilePath( char* fspec, cha
  => r* p, int flag )
bool              FindFilePath          ::findFirstPrimitive()
bool              FindFilePath          ::findFirst()
bool              FindFilePath          ::findNext()



                                                  ::onCommCheckMsg( HWND, Comm* cm
void              DlgWin
  =>  )                                           ::onCommSendSyncMsg( HWND, Comm*
void              DlgWin
  =>  cm )                                        ::onSpeech( HWND hwnd, SD_CHANNE
void              DlgWin
  => L ch )                                       ::onSysCommand( HWND hwnd, UINT
void              DlgWin
  => cmd, int, int )                              ::onCommand( HWND hwnd, UINT cmd
void              DlgWin
  => , HWND, UINT )                               ::onTimerDrawVU( HWND hwnd, UINT
void              DlgWin
  =>  id )                                        ::onDestroy( HWND hwnd )
void              DlgWin                          ::itsBad( HWND hwnd, UttChannel*
void              DlgWin
  =>  toMe )                                      ::itsGoodAdapt( CityHypothesis&
void              DlgWin
  => ch )                                         ::buildCityHypothesis()
void              DlgWin                          ::outCommPort( const char* zp, c
void              DlgWin
  => onst char* st, const char* cty )             ::reportError( int code, char fa
void              DlgWin
  => r *message )
```

WAPP.CPP 7-22-95 11:36a

```
bool              DlgWin                          ::onInitDialog( HWND hDlg, HWND,
  =>  long )                                      ::isCancelBadLabelGoToSleep( HWN
bool              DlgWin
  => D hwnd )                                     ::isCertain( SD_WORD testWord )
bool              DlgWin                          ::isCancelBadLabelGoToSleepOrPro
bool              DlgWin
  => vince( HWND hwnd )

CityHypothesis* DlgWin                     ::rerecogZip( int* foundDistance )
void SD_CALLBACK _export DlgWin ::postSpeechEvent( SD_CHANNEL ch, SD_CHANNEL_EVE
  => NT eventIn )
BOOL FAR PASCAL  _export DlgWin ::wndProc( HWND hwnd, UINT message, WPARAM wPara
  => m, LPARAM lParam )



                                                  ::onCommand( HWND hWnd, UINT cmd
void              AppWin
  => , HWND, UINT )                               ::onDestroy( HWND )
void              AppWin                          ::onNCDestroy( HWND )
void              AppWin                          ::onClose( HWND hWnd )
void              AppWin                          ::onCreate( HWND hwnd, CREATESTR
bool              AppWin
  => UCT FAR* )

long FAR PASCAL  _export AppWin  ::wndProc( HWND hwnd, UINT message, WPARAM wPara
  => m, LPARAM lParam )



        2. Helper functions, which do not belong to a class


                                                  setSpecialZip( CustomCitySL* sl,
static char*
  =>  char* value )                               setDefaultCommPars( char* value
static char*
  => )                                            setPureState( char* value )
static char*                                      setStateAlias( char* value )
static char*                                      setUserPath( char* value )
static char*                                      setFepPar( WORD id, char* value
static char*
  => )                                            setStartOfSpeech( char* value )
static char*                                      setEndOfSpeech( char* value )
static char*                                      setNothing( char* )
static char*                                      setCommand( char* var )
static char*                                               parseConfig( const char*
void
  =>  filename )                                           formatOutput( char* outB
void
  => uf, const char* format,



        3. Main function

int PASCAL WinMain( HINSTANCE hinst, HINSTANCE hPrevInstance, LPSTR lpszCmdLine,
  =>  int nCmdShow )



   CLASSES declared  in WAPP.CPP
   .............................


        class DlgWin
```

Page 1 of 29

```
        class ChoiceListStats
        class CustomCitySL                              : public CitySL
        class WordIdToCustomCityHS          : public HS< CustomCitySL >
        class FindFilePath
        class AppWin

        struct ConfigCommands

                                                                              */
  =>
////////////////////////////////////////////////////////////////////////////////



// INCLUDE


#include <windowsx.h>
#include <string.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <bios.h>
#include <io.h>
#include <stdio.h>
#include <fcntl.h>
#include "dragcpp.h"


extern "C" {
    #include "fepext.h"
}

#include "wapp.h"
#include "slidectl.h"


#include <mmsystem.h>
#include <ddeml.h>


extern long computeMemoryUse();


// This is the instance of the DgnSlideControl

DgnSlideControl         slideControl;


extern HINSTANCE        hInstance;      // when we call DgnSlideControl::libMain
  => Init()
HINSTANCE                      hInstance;      // hInstance must be initialized
  => ...


#include "hashcsz.h"
#include "deamon.h"


 // DEFINE
```

```
#define WAKEUP_STATE                          0
#define ZIP_STATE                                   1
#define CITY_STATE                                  2

#define FEP_STARTOFSPEECHTHRESH 13
#define FEP_ENDOFOSPEECHTHRESH  12

#define handleMessage( message, hWnd, wParam, lParam, fn ) \
 case (message) : returnValue = HANDLE_ ## message( (hWnd), (wParam), (lParam),
  => (fn) ); break;



// STATIC VARIABLES. Names of vocabularies are defined here.

static char femCityVoc[]         = "ctall.vof";
static char femDigitsVoc[]       = "digits.vof";
static char malCityVoc[]         = "ctall.vom";
static char malDigitsVoc[]       = "digits.vom";
static char* cityVoc = 0;
static char* digitsVoc = 0;

static char origCityUser[ 128 ];
static char tmpCityUsr[] = "tmpcty.usr";
static char* cityUser = origCityUser;

static char userPathString[ 256 ] = ".";

static char bluePrefix[ 6 ] = "00";
static char orangePrefix[ 6 ] = "09";

static char lastZipOutCommPort[ UTT_PROMPT_LENGTH ] = "";
static char lastStateOutCommPort[ UTT_PROMPT_LENGTH ] = "";
static char lastCityOutCommPort[ UTT_PROMPT_LENGTH ] = "";

extern char stateAliasString[];



static char* sayPrompt[] = {
        "Say: WakeUp",
        "Say: ZipState",
        "Say: City",
};



//////////////////////////////////////////////////////////////////////////////
//
//      CLASS ChoiceListStats
//
//      Purpose: tracks results recognizer finds and the recognition errors.
//  Info used to update confusability matrix. Can be turned off later.
//
//////////////////////////////////////////////////////////////////////////////


class ChoiceListStats
{
 protected:

        int digitStats[ 500 ];
        char openRecResult[ 128 ];

 public:
        void setOpenRecResult( const char* src ) { strcpy( openRecResult, src );
  => }

        void recordResult( const char* src );

        void printStats();
};


void ChoiceListStats::recordResult( const char* src )
{
        char *o = openRecResult;

        for( int i = 0; i < 5; ++i )
        {
                if( !isdigit( *o ) || !isdigit( *src ) )
                        return;

                digitStats[ i*100 + 10*( *src - '0' ) + ( *o - '0' ) ] += 1;

                if( *++o && *++src == ' ' )
                        ++src;

                else
                        return;
        }
}

// Print statistics to file

void ChoiceListStats::printStats()
{
        FILE *out = fopen( "digstats", "a+" );

        if( out )
        {
                int *v = digitStats;

                fprintf( out, "///////////////////////////////////////////\n" );

                for( int i=0; i<5; ++i )
                {
                        fprintf( out, "// %d\n", i );

                        for( int j=0; j<10; ++j )
                        {
                                fprintf( out, "%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%
 => d\t%d\t\n",
                                                                v[0], v[1], v[2
 => ], v[3], v[4],
                                                                v[5], v[6], v[7
 => ], v[8], v[9] );
                                v += 10;
                        }
                }

                fclose( out );

                memset( openRecResult, 0, sizeof( openRecResult ) );
        }
```

WAPP.CPP 7-22-95 11:36a

```
}

// declare class for the statistics
static ChoiceListStats choiceListStats;



//////////////////////////////////////////////////////////////////////////////
//
//      CLASS DlgWin
//
//
//      Purpose: Drives the application.
//
//              The member function onSpeech deals with the recognition
//              of the zip code, state name and city name. It applies
//              correction mechanisms when the proposed answers of the three
//              components do not lock tog ether.
//
//////////////////////////////////////////////////////////////////////////////


class DlgWin
{
 public:
        static BOOL FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );
        static HWND      hWnd;

 protected:

        // VARIABLES
        static bool     messageHandled;
        static bool     sawError;
        static bool     ignoreErrors;
        static bool     fixError;
        static bool     wasListening;

        static char*    name;
        static short    wordPause;
        static short    startThresh;
        static short    endThresh;

        static int              appState;                       // zips, city, s
  => tate, error
        static long     numErrors;


        static char digitSpelling[ UTT_PROMPT_LENGTH ];

        // App specific CLASSES and STRUCTURES
        static AC< CityHypothesis > cityHypo;
        static AC< long > zips;

        static ZipResult        zipResult;
        static ZipHypoAC        zipHypo;
        static Comm*            comm[4];


        // DRAGON recognizer variables
        static SpeechTask*      speechTask;             // speech driver task ID
        static UttChannel*      channel;
        static Recognizer*      recog;
        static UttChannel       zipUtt, cityUtt;
```

Page 3 of 29

```
        static SD_WORD  cancelWord;
        static SD_WORD  badLabelWord;
        static SD_WORD  goToSleepWord;
        static SD_WORD  wakeUpWord;
        static SD_WORD  sameZipWord;
        static SD_STATE testState;

        // scratch buf
        static char buf[ UTT_PROMPT_LENGTH ];


        // FUNCTIONS
        static void reportError( int code, char far *message );
        static void SD_CALLBACK _export postSpeechEvent( SD_CHANNEL, SD_CHANNEL_
  => EVENT );

        // wndProc() services
        static bool onInitDialog( HWND hWnd, HWND topCntrl, long );
        static void onDestroy( HWND hWnd );
        static void onSysCommand( HWND hWnd, UINT cmd, int x, int y );
        static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y );

        // core wndProc function
        static void onSpeech( HWND hWnd, SD_CHANNEL ch );

        static void onCommCheckMsg( HWND hWnd, Comm* cm );
        static void onCommSendSyncMsg( HWND hWnd, Comm* cm );
        static void onTimerDrawVU( HWND hwnd, UINT id );


        // sub service routines for the main winProc services
        static bool isGoodZipPhrase();
        static bool isCancelBadLabelGoToSleepOrProvince( HWND );
        static bool isCancelBadLabelGoToSleep( HWND );
        static void itsGoodAdapt( CityHypothesis& );
        static void itsBad( HWND, UttChannel*  );
        static CityHypothesis* rerecogZip( int* foundDistance );
        static void outCommPort( const char* zp, const char* st, const char* cty
  =>  );
        static void buildCityHypothesis();

        static bool isCertain( SD_WORD );

        friend int PASCAL WinMain( HINSTANCE, HINSTANCE, LPSTR, int );
};


////////////////////////////////////////////////////////////////////////////
//
//      DlgWin variables and classes defined in the DlgWin Class are
//      made global.
//
////////////////////////////////////////////////////////////////////////////


HWND            DlgWin::hWnd;
bool            DlgWin::messageHandled;
SpeechTask*     DlgWin::speechTask;
Recognizer*     DlgWin::recog;                  // recognizer, wrapper around SD
  => API (dragcpp.*)
UttChannel*     DlgWin::channel;
bool            DlgWin::sawError;
bool            DlgWin::ignoreErrors;
```

```
char*           DlgWin::name = "dragon123\0      ";

UttChannel      DlgWin::zipUtt;
UttChannel      DlgWin::cityUtt;

ZipHypoAC       DlgWin::zipHypo( 64 );

AC< CityHypothesis >    DlgWin::cityHypo( 64 );
AC< long >      DlgWin::zips;

ZipResult       DlgWin::zipResult;
char            DlgWin::digitSpelling[ UTT_PROMPT_LENGTH ];

bool            DlgWin::wasListening;
bool            DlgWin::fixError = 1;
long            DlgWin::numErrors;
short           DlgWin::wordPause;
short           DlgWin::startThresh;
short           DlgWin::endThresh;


int                     DlgWin::appState;                       // zips, city, s
  => tate, error
SD_WORD         DlgWin::cancelWord = 0;
SD_WORD         DlgWin::badLabelWord = 0;
SD_WORD         DlgWin::goToSleepWord = 0;
SD_WORD         DlgWin::wakeUpWord = 0;
SD_WORD         DlgWin::sameZipWord = 0;
SD_STATE        DlgWin::testState;


char            DlgWin::buf[ UTT_PROMPT_LENGTH ];       // scratch buf
Comm*           DlgWin::comm[4];



////////////////////////////////////////////////////////////////////////
//                                      configuration file stuff
//
//      Table that contains the names of the functions and the pointers
//      to the functions that can be called in the configuration file
//
////////////////////////////////////////////////////////////////////////


struct ConfigCommands
{
        const char* name;
        char* (*command)( char* );
};


static char* setCommand( char* );


static ConfigCommands cfgCommands[] = {
        { "set", setCommand },
        { 0, 0 }
};
```

```
///////////////////////////////////////////////////////////////////////
//
// CLASS CustomCitySL
//
// Purpose:
//
//      Process Command Words and Provinces
//
///////////////////////////////////////////////////////////////////////

class CustomCitySL : public CitySL
{
 public:
        SD_WORD wordId;
        bool isControlWord;

        CustomCitySL( long z, char* n, SD_WORD id, bool isCntrl )
        : CitySL( z, n ) { wordId = id; isControlWord = isCntrl; }
};


///////////////////////////////////////////////////////////////////////
//
// CLASS WordIdToCustomCityHS
//              Inherits from CustomCitySL, Hash Table
//
// Purpose:
//
//              Take word ID and find zip code (dummy for those special provinc
  => es
//              internal to mercury.
//
///////////////////////////////////////////////////////////////////////


class WordIdToCustomCityHS : public HS< CustomCitySL >
{
 protected:

        unsigned keyHash( const SLinkBase* ) const;

 public:
        WordIdToCustomCityHS( unsigned maxSize = 32, unsigned initialSize = 32 )
        : HS< CustomCitySL >( maxSize, initialSize ) { }

        int compare( const SLinkBase*, const SLinkBase* ) const;
};



unsigned WordIdToCustomCityHS::keyHash( const SLinkBase* slb ) const
{
        return ((CustomCitySL* )slb)->wordId;
}


int WordIdToCustomCityHS::compare( const SLinkBase* slb1, const SLinkBase* slb2
  => ) const
{
        return( ( (CustomCitySL* )slb1 )->wordId -
```

```
                                          ( (CustomCitySL* )slb2 )->wordId );
}

static WordIdToCustomCityHS zipCommandWordsHS;


/////////////////////////////////////////////////////////////////////////////
//
//
// set a variable... the particular data and functions known to setCommand()
//
//
/////////////////////////////////////////////////////////////////////////////


// Some general commands
static CustomCitySL badLabelSL(                  1, "badlabel",
  =>          0, 1 );
static CustomCitySL cancelAddressSL(             2, "canceladdress",
  =>          0, 1 );
static CustomCitySL sameZipSL(                   3, "samezip",
  =>          0, 1 );
static CustomCitySL goToSleepSL(                 4, "gotosleep",
  =>          0, 1 );
static CustomCitySL wakeUpSL(                    5, "wakeup",
  =>          0, 1 );


// state specific rules
static CustomCitySL albertaSL(                20, "alberta",
  =>          0, 0 );
static CustomCitySL britishColumbiaSL(        21, "britishcolumbia",
  => 0, 0 );
static CustomCitySL labradorSL(               22, "labrador",
  => 0, 0 );
static CustomCitySL manitobaSL(               23, "manitoba",
  => 0, 0 );
static CustomCitySL newBrunswickSL(           24, "newbrunswick",
  => 0, 0 );
static CustomCitySL newFoundlandSL(           25, "newfoundland",
  => 0, 0 );
static CustomCitySL northWestTerritoriesSL( 26, "northwestterritories", 0, 0 );
static CustomCitySL novaScotiaSL(             27, "novascotia",
  => 0, 0 );
static CustomCitySL ontarioSL(                28, "ontario",
  =>          0, 0 );
static CustomCitySL princeEdwardIslandSL(     29, "princeedwardisland",   0, 0 );
static CustomCitySL quebecSL(                 30, "quebec",
  =>          0, 0 );
static CustomCitySL saskatchewanSL(           31, "saskatchewan",
  => 0, 0 );
static CustomCitySL yukonTerritorySL(         32, "yukonterritory",
  => 0, 0 );

static CustomCitySL mexicoSL(                 40, "mexico",
  =>          0, 0 );



static char* setSpecialZip( CustomCitySL* sl, char* value )
{
        int i = 0;
        for( char *s=value; *s; ++s )
        {
                i += ( *s >= '0' && *s <= '9' );
        }

        if( i == 0 )
                return "Bad special Zip code assingment";

        long test = atol( value );

        sl->zip = test;

        return 0;
}



static char* setBadLabelZip( char* value )              { return setSpecialZip(
  => &badLabelSL, value ); }
static char* setCancelAddressZip( char* value ) { return setSpecialZip( &cancelA
  => ddressSL, value ); }
static char* setSameZipZip( char* value )               { return setSpecialZip(
  => &sameZipSL, value ); }
static char* setGoToSleepZip( char* value )             { return setSpecialZip(
  => &goToSleepSL, value ); }
static char* setWakeUpZip( char* value )                { return setSpecialZip(
  => &wakeUpSL, value ); }

static char* setAlbertaZip( char* value )               { return setSpecialZip(
  => &albertaSL, value ); }
static char* setLabradorZip( char* value )              { return setSpecialZip(
  => &labradorSL, value ); }
static char* setManitobaZip( char* value )              { return setSpecialZip(
  => &manitobaSL, value ); }
static char* setNewBrunswickZip( char* value )  { return setSpecialZip( &newBrun
  => swickSL, value ); }
static char* setNewFoundlandZip( char* value )  { return setSpecialZip( &newFoun
  => dlandSL, value ); }
static char* setNovaScotiaZip( char* value )    { return setSpecialZip( &novaSco
  => tiaSL, value ); }
static char* setOntarioZip( char* value )               { return setSpecialZip(
  => &ontarioSL, value ); }
static char* setQuebecZip( char* value )                { return setSpecialZip(
  => &quebecSL, value ); }
static char* setSaskatchewanZip( char* value )  { return setSpecialZip( &saskatc
  => hewanSL, value ); }
static char* setYukonTerritoryZip( char* value ){ return setSpecialZip( &yukonTe
  => rritorySL, value ); }
static char* setMexicoZip( char* value )                { return setSpecialZip(
  => &mexicoSL, value ); }
static char* setBritishColumbiaZip( char* value )                   { return setSpec
  => ialZip( &britishColumbiaSL, value ); }
static char* setNorthWestTerritoriesZip( char* value )  { return setSpecialZip(
  => &northWestTerritoriesSL, value ); }
static char* setPrinceEdwardIslandZip( char* value )    { return setSpecialZip(
  => &princeEdwardIslandSL, value ); }



static CommPars* comPars[ 4 ];
static char* commFormat[ 4 ];


// Helper function
```

```
static char* setDefaultCommPars( char* value )
{
        for( char *s = value; *s; ++s )
                if( *s == ',' )   *s = ' ';

        char parityStr[ 10 ], comPort[ 20 ], formatString[ 128 ];

        int baud, dataBits, stopBits;

        int numFields = sscanf( value, "%s %d %s %d %d %s",

  =>                            comPort, &baud, parityStr, &dataBits, &stopBits, fo
  => rmatString );
        if( numFields < 5 )
                return "wrong number/order of arguments... ex:\n\tset comm = com
  => 1 9600 n 8 1";

        // check to see if the arguments are reasonable and set the port...

        int port=comPort[ 3 ] - '0';

        if( 0>port || port>4 )
                return "bad commPort number, (must be 1 thru 4)";

        if( baud != 19200 && baud != 14400 && baud != 9600 &&
                        baud != 4800 && baud != 2400 && baud != 1200 )
                return "unknown commPort Baud rate";

        int parity;

        switch( tolower( *parityStr ) )
        {
         case 'n':      parity = NOPARITY;      break;
         case 'o':      parity = ODDPARITY;     break;
         case 'e':      parity = EVENPARITY;    break;
         case 'm':      parity = MARKPARITY;    break;
         case 's':      parity = SPACEPARITY;   break;

         default:
                return "unknown commPort Perity";
        }

        if( dataBits != 7 && dataBits != 8 )
                return "strange number of commPort dataBits";

        if( stopBits == 1 )
                stopBits = ONESTOPBIT;

        else if( stopBits == 2 )
                stopBits = TWOSTOPBITS;

        else
                return "strange number of commPort stopBits";

        if( numFields == 5 )
                commFormat[ port-1 ] = 0;

        else
        {
                commFormat[ port-1 ] = new char[ strlen( formatString ) + 1 ];

                strcpy( commFormat[ port-1 ], formatString );
        }

        comPars[ port-1 ] = new CommPars( port, baud, parity, dataBits, stopBits
  => );
```

```
        return 0;
}

// Helper function
static char* setPureState( char* value )
{
        for( char *s = value; *s && !isspace( *s );     ++s );

        *s = '\0';

        StateZip* sz = StateZip::findStateZip( value );

        if( sz == 0 )
                return "bad Pure State in UPS.CFG";

        sz->pure( TRUE );

        return 0;
}

// Helper function

static char* setStateAlias( char* value )
{
        for( char *s = value; *s && !isspace( *s );     ++s );

        *s = '\0';

        StateZip* sz = StateZip::findStateZip( value );

        if( sz == 0 )
                return "in Ups.cfg stateAlias, incorrect state spelling (must be
  =>  lowercase and one word)";

        sz->pure( TRUE );

        return 0;
}
// Helper function

static char* setUserPath( char* value )
{
        for( char *s = value; *s && !isspace( *s );     ++s );

        *s = '\0';

        strcpy( userPathString, value );

        return 0;
}


// Helper function
static char* setFepPar( WORD id, char* value )
{
        for( char *s = value; *s && !isspace( *s );     ++s );
```

```
        *s = '\0';

        if( *value )
        {
                WORD v;

                v = (WORD )atol( value );

                Fep_Setpar( SUPER_USER, id, v );

                return 0;
        }
        return "Par Not Set";
}


// Helper function

static char* setStartOfSpeech( char* value )
{
        if( setFepPar( FEP_STARTOFSPEECHTHRESH, value ) )
                return "StartOfSpeech Not Set";

        return 0;
}

// Helper function

static char* setEndOfSpeech( char* value )
{
        if( setFepPar( FEP_ENDOFOSPEECHTHRESH, value ) )
                return "EndOfSpeech Not Set";

        return 0;
}


// Helper function

static char* setNothing( char* )
{
        return 0;
}


static ConfigCommands setCommands[] = {
        { "Com",                                setDefaultCommPars
  =>                    },
        { "Comm",                               setDefaultCommPars
  =>                    },
        { "ComPort",                    setDefaultCommPars
  =>            },
        { "Pure",                               setPureState
  =>                    },
        { "UserPath",                   setUserPath
  =>            },
        { "StartOfSpeech",              setStartOfSpeech
  =>            },
        { "EndOfSpeech",                setEndOfSpeech
  =>            },

        { "ZipRange",                   setNothing
  =>                    },
        { "BadLabel",                   setBadLabelZip
 =>         },

        { "Alberta",                    setAlbertaZip
 =>         },
        { "BritishColumbia",    setBritishColumbiaZip               },
        { "Labrador",                   setLabradorZip
 =>         },
        { "Manitoba",                   setManitobaZip
 =>         },
        { "NewBrunswick",               setNewBrunswickZip
 =>         },
        { "Newfoundland",               setNewfoundlandZip
 =>         },
        { "NorthWestTerritories",       setNorthWestTerritoriesZip      },
        { "NovaScotia",                 setNovaScotiaZip
 =>         },
        { "Ontario",                    setOntarioZip
 =>         },
        { "PrinceEdwardIsland", setPrinceEdwardIslandZip            },
        { "Quebec",                             setQuebecZip
 =>                 },
        { "Saskatchewan",               setSaskatchewanZip
 =>         },
        { "YukonTerritory",             setYukonTerritoryZip
 => },

        { "Mexico",                             setMexicoZip
 =>                 },

        { 0, 0 }
};

static char* setCommand( char* var )
{
        while( isspace( *var ) )
                ++var;

        char* value = var;

        while( isalnum( *value ) || *value == '_' )
                ++value;

        if( *value == '\0' )
                return "no value found in set command... ex: set variable = valu
 => e";

        bool foundEqual = (*value == '=');

        *value++ = '\0';

        if( !foundEqual )
        {
                while( isspace( *value ) )
                        ++value;

                if( *value != '=' )
                        return "no \"=\" sign in set command... ex: set variable
 =>  = value";

                ++value;
        }

        while( *value && isspace( *value ) )
```

```
                ++value;

        for( int i=0; setCommands[ i ].name != 0; ++i )
        {
                if( stricmp( var, setCommands[ i ].name ) == 0 )
                        return setCommands[ i ].command( value );
        }

        return "unknown set variable";
}

// Helper function

void parseConfig( const char* filename )
{
        FILE *cfgfile = fopen( filename, "r" );

        char *errorMessage = ( cfgFile == 0

 =>                     ? "could not open UPS.CFG file"

 =>                     : 0 );
        char line[ 256 ];
        int lineNumber = 0;

        while( errorMessage == 0 && fgets( line, sizeof( line ), cfgFile ) != 0
 => )
        {
                char *var = line;

                ++lineNumber;

                while( isspace( *var ) )
                        ++var;

                if( *var == '#' || *var == '\0' )
                        continue;

                char *value = var;

                while( !isspace( *value ) && *value != '\0' )
                        ++value;

                if( *value != '\0' )
                {
                        *value++ = '\0';

                        while( isspace( *value ) )
                                ++value;
                }

                for( int i=0; cfgCommands[ i ].name != 0; ++i )
                {
                        if( stricmp( var, cfgCommands[ i ].name ) == 0 )
                        {
                                errorMessage = cfgCommands[ i ].command( value )
 => ;

                                break;
                        }
                }
        }

        if( errorMessage )
        {
```

```
		char errorType[ 40 ] = "UPS.CFG error, line            \n";

		itoa( lineNumber, errorType + 20, 10 );

		MessageBox( DlgWin::hWnd, errorMessage, errorType, MB_OK | MB_IC
  => ONEXCLAMATION );
	}

	if( cfgFile )
		fclose( cfgFile );
}



//////////////////////////////////////////////////////////////////////
//
// class FindFilePath
//
// Purpose:
//
//		Find file using PATH variable
//
//////////////////////////////////////////////////////////////////////


class FindFilePath
{
 protected:

	static char defaultPath[];
	static char defaultFileSpec[];

	char *pathString, *path;
	char *fileSpec;
	int attribFlag;

	struct ffblk attrib;

	bool findFirstPrimitive();

 public:

	FindFilePath( char* fspec, char* p =0, int flag =FA_NORMAL );

	~FindFilePath()
	{
		if( pathString != defaultPath )
			delete pathString;

		if( fileSpec != defaultFileSpec )
			delete fileSpec;
	}

	bool FindFilePath::findFirst();
	bool FindFilePath::findNext();

	char* filename() { return attrib.ff_name; }
};

char FindFilePath::defaultFileSpec[] = "*.*";
char FindFilePath::defaultPath[] = ".\0";
```

```
// Constructor

FindFilePath::FindFilePath( char* fspec, char* p, int flag )
{
	attribFlag = flag;

	if( p )
	{
		path = pathString = new char[ strlen( p ) + 2 ];
		strcpy( pathString, p );
		for( p = pathString; *p; ++p )
			if( *p == ';' )
				*p = '\0';

		*++p = '\0';
	}
	else
		pathString = path = defaultPath;

	if( fspec )
	{
		fileSpec = new char[ strlen( fspec ) + 1 ];
		strcpy( fileSpec, fspec );
	}
	else
		fspec = defaultFileSpec;
}

// Helper function

bool FindFilePath::findFirstPrimitive()
{
	for(;;)
	{
		if( *path == '\0' )
			return FALSE;

		char* fSpec = new char[ strlen( path ) + strlen( fileSpec ) + 2
  => ];

		char* p = stpcpy( fSpec, path );

		if( p>fSpec && *(p-1) != ':' && *(p-1) != '\\' && *(p-1) != '/'
  => )
			*p++ = '/';

		p = stpcpy( p, fileSpec );

		path +=  + strlen( path ) + 1;

		if( !findfirst( fSpec, &attrib, attribFlag ) )
		{
			delete fSpec;
			return TRUE;
		}
		delete fSpec;
	}
}

// Helper function

bool FindFilePath::findFirst()
{
	path = pathString;

	return findFirstPrimitive();
```

```
}

// Helper function

bool FindFilePath::findNext()
{
        if( !findnext( &attrib ) )
                return TRUE;

        return findFirstPrimitive();
}


// initialisation function for DlgWin class (NOT constructor)
// the global variable appState is set to ZIP_STATE


bool DlgWin::onInitDialog( HWND hDlg, HWND, long )
{
        hWnd = hDlg;

        speechTask = new SpeechTask( name, reportError );


        //              Memory debug stuff

        //      int16 track;
        //
        //      speechTask->getPar( "track-mem", &track );
        //
        //      speechTask->setPar( "track-mem", (short )!track );

        recog = 0;

        StateZip::initZipToUseStateZip();

        //      Fep_Setpar( SUPER_USER, 13, 87 );
        //      Fep_Setpar( SUPER_USER, 12, 58 );

        parseConfig( "ups.cfg" );

        for( int i=0; i<4; ++i )
        {
                if( comPars[ i ] != 0 )
                {
                        comm[ i ] = new Comm( hDlg, comPars[ i ] );

                        if( comm[ i ]->error() != 0 )
                        {
                                MessageBox( hDlg, "Comm Error", "Port Initialize
  => tion failed", MB_OK | MB_ICONEXCLAMATION );

                                delete comm[ i ];

                                comm[ i ] = 0;
                        }

                        delete comPars[ i ];
                }
        }

        FindFilePath ffPath( "*.un?", userPathString );

        // set up the usr listbox
```

WAPP.CPP 7-22-95 11:36a

```
        if( ffPath.findFirst() )
        {
                do
                {
                        SendDlgItemMessage( hDlg, USR_COMBO_ID, CB_ADDSTRING, 0,
=>  (DWORD) ffPath.filename() );
                }
                while( ffPath.findNext() );
        }


        appState = ZIP_STATE;

        int16 compttn;
        //      speechTask->getPar( "computation", &compttn );
        speechTask->setPar( "computation", (short )2 );
        //      speechTask->setPar( "adapt-tolerance", (short )3 );

        SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLRANGE, 0, 0x0
=> 1000060L );
        SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLPOS, 0x00a0,
=> 1 );

        SendDlgItemMessage( hDlg, START_THRESH_SLIDE_ID, WM_SETSCROLLRANGE, 0, 0
=> x00800020L );
        SendDlgItemMessage( hDlg, START_THRESH_SLIDE_ID, WM_SETSCROLLPOS, 0x0050
=> , 1 );

        SendDlgItemMessage( hDlg, END_THRESH_SLIDE_ID, WM_SETSCROLLRANGE, 0, 0x0
=> 0700010L );
        SendDlgItemMessage( hDlg, END_THRESH_SLIDE_ID, WM_SETSCROLLPOS, 0x0030,
=> 1 );

        char line [100];
        char spelling[50];

        for( i=0; i<100; ++i )
        {
                char zipFileName[] = "zip/zip.000000000";

                itoa( i, zipFileName + 8 + (i<10), 10 );

                FILE* file = fopen( zipFileName, "r" );

                if( file == 0 )     continue;

                fclose( file );

                StateZip::initHSs( zipFileName );
        }

        return TRUE;
}


// Helper function

void formatOutput( char* outBuf, const char* format,                    const c
  => har* zip, const char* state, const char* city )
{
        char* buf = outBuf;

        while( *format )
        {
                switch( *format )
```

Page 10 of 29

```
	{
	case 'X':
		++format;
		if( tolower( *format ) == 'z' )
		{
			const char* z=zip;
			for( int i=0; i<5 && (*buf = *z++) != 0; ++i )
				++buf;

			++format;
		}
		else if( tolower( *format ) == 's' )
		{
			const char* s=state;
			while( (*buf = *s++) != 0 )
				++buf;

			++format;
		}
		else if( tolower( *format ) == 'c' )
		{
			const char* c=city;
			while( (*buf = *c++) != 0 )
				++buf;

			++format;
		}
		else
			*buf++ = 'X';
		break;

	case '\\':
		++format;

		if( tolower( *format ) == 'r' )
		{
			*buf++ = '\r';

			++format;
		}
		else if( tolower( *format ) == 'n' )
		{
			*buf++ = '\n';

			++format;
		}
		else if( tolower( *format ) == 'b' )
		{
			*buf++ = ' ';

			++format;
		}
		else if( tolower( *format ) == 't' )
		{
			*buf++ = '\t';

			++format;
		}
		else
		{
			if( *format != '\0' )
			{
				*buf++ = *format++;
			}
		}
		break;
     WAPP.CPP  7-22-95  11:36a
```

```
		default:
			*buf++ = *format++;
		}
	}

	*buf = 0;

	assert( buf - outBuf < 256 );
}


// function that sends the recognition answer found in onSpeech to the
// device via a comm port.

void DigWin::outCommPort( const char* zp, const char* st, const char* cty )
{
	channel->beep( "ding.wav" );

	strcpy( lastZipOutCommPort, zp );
	strcpy( lastStateOutCommPort, st );
	strcpy( lastCityOutCommPort, cty );

	char outBuf[ 256 ];

	for( int i=0; i<4; ++i )
	{
		if( comm[ i ] )
		{
			formatOutput( outBuf, commFormat( i ) ? commFormat( i )
 => : "%z\r",

 => zp, st, cty );

			comm[ i ]->comWrite( outBuf, strlen( outBuf ) );
		}
	}

	static long lastInterval = 0;
	static time_t lastTime = 0;

	long currentTime = biostime( 0, 0 );

	lastInterval = (lastInterval + (currentTime - lastTime)) / 2;

	if( (unsigned long)lastInterval > 1000 )
		lastInterval = 1000;

	long rate = (3600L * 1000L / 55L) / lastInterval;

	lastTime = currentTime;

	ltoa( rate, outBuf, 10 );

	char *s = outBuf + strlen( outBuf );

	strcpy( s, " pkgs/hour      " );

	SetWindowText( GetDlgItem( hWnd, STAT1_TEXT_ID ), outBuf );

	static long totalNumberOfPackages = 0;

	ltoa( ++totalNumberOfPackages, outBuf, 10 );

	s = outBuf + strlen( outBuf );
                                   Page 11 of 29
```

```
        strcpy( s, "  pkgs       " );

        SetWindowText( GetDlgItem( hWnd, STAT2_TEXT_ID ), outBuf );

        ltoa( numErrors, outBuf, 10 );

        s = outBuf + strlen( outBuf );

        strcpy( s, "  BadLabels       " );

        SetWindowText( GetDlgItem( hWnd, STAT3_TEXT_ID ), outBuf );
}

// Dragon speech driver error handler.
// Immediately after having

void DlgWin::reportError( int code, char far *message )
{
        sawError = YES;
        if( ignoreErrors )
                return;         // prevent recursion
        ignoreErrors = YES;

        // Throw up an error message
        char buffer[ 512 ];
        sprintf( buffer, "An error has occurred\n  code = %d\n  message = %s",
                                        code, message );

        //    if( stricmp( message, "Bad token" ) && strncmp( message, "Invalid
  => word handle:", 18 ) )
        //    {
        MessageBox( hWnd, buffer, "Wintalk SDAPI Error",
                                                        MB_TASKMODAL | MB_ICONST
  => OP | MB_OK );

        // terminate...  kill the hTopParentWindow
        //if( hWnd )
        //    PostMessage( hWnd, WM_CLOSE, 0, 0 );
        //    }
        ignoreErrors = 0;
}

//////////////////////////////////////////////////////////////////////////////
/* postSpeechEvent()

Callback function, handed to the microphone channel when it was opened.

We do not process any data in this function. We want to get out of this
interrupt as soon as possible.

Its place in the control flow:

Speech --> Board digitizes --> Multimedia (MM) layer discovers speech, and
interrupts other Windows activities --> dragdev, the dll that deals
with the front end (FEP), gets data from MM and decides whether we have
speech or not(Utterance detection).
If so, dragdev stores the utterance in its queue and it also calls
postSpeechEvent. This function was handed to the channel when we opened
the channel. In this way the microphone knows to use this callback function
to get into the application for recognition.

After we leave the interrupt we send ourselves a PostMessage UM_ChannelStart.
When UM_ChannelStart is processed we direct the application to the function
```

```
onSpeech().

Make sure there is a start of utterance message for any utterance.
It was observed that having utterance and the related utterance start
message were out of sync. If PeekMessage return false, we will
make sure to put a UM_CHANNEL_START message in. This makes it more robust
to keep he number of utterances and CHANNEL_STARTS in sync.

A similar trick is applied in onSpeech() at EndOfSwitch.
*/
//////////////////////////////////////////////////////////////////////////////


void SD_CALLBACK _export DlgWin::postSpeechEvent( SD_CHANNEL ch, SD_CHANNEL_EVEN
  => T eventIn )
{
        assert( eventIn == SD_CHANNEL_START );

        MSG msg;

        if( 0 == PeekMessage( &msg, hWnd, UM_CHANNELSTART, UM_CHANNELSTART, PM_N
  => OREMOVE | PM_NOYIELD ) )
                //??
                PostMessage( hWnd, UM_CHANNELSTART, (WORD )ch, 0 );

} // EventHandler



#define HANDLE_UM_CHANNELSTART(hwnd, wParam, lParam, fn) \
((fn)(hwnd, (SD_CHANNEL )wParam), 1L)


#define HANDLE_UM_COMM_CHECK_MSG( hwnd, wParam, lParam, fn ) \
((fn)(hwnd, (Comm* )lParam), 1L)


#define HANDLE_UM_COMM_SEND_SYNC_MSG( hwnd, wParam, lParam, fn ) \
((fn)(hwnd, (Comm* )lParam), 1L)

// helper function

void DlgWin::onCommCheckMsg( HWND, Comm* cm )
{
        assert( cm != 0 );
        assert( cm == comm[0] || cm == comm[1] || cm == comm[2] || cm == comm[3]
  => );

        cm->checkMessage();
}


// helper function

void DlgWin::onCommSendSyncMsg( HWND, Comm* cm )
{
        assert( cm != 0 );
        assert( cm == comm[0] || cm == comm[1] || cm == comm[2] || cm == comm[3]
  => );

        cm->sendSyncMsg();
}
```

```
static long right=0, wrong=0, predicted=0, addressConfidence=100;


extern const char* ltoZip( long num );


/////////////////////////////////////////////////////////////////////////
//
//      Given a zip and a state, build a list with zip state and cities
//
//
//
/////////////////////////////////////////////////////////////////////////


void DigWin::buildCityHypothesis()
{
        // build bigger city list here....
        cityHypo.removeAll();


        // number of words on the choice list after discrete recognition
        int i, j, k, cnt = zipUtt.resultCount();

        if( cnt > 6 )
                cnt = 6;

        ZipHypothesis test[ 6 ];

        for( i=0, k=0; i<cnt; ++i )
        {
                SD_WORD t = zipUtt.resultId( i );

                for( j=0; j<zipHypo.count(); ++j )
                {
                        if( t == zipHypo[ j ].wordId )
                        {
                                new( (void* )&test[ k++ ] )  ZipHypothesis( zipH
 => ypo[ j ] );

                                break;
                        }
                }

                //      if( j == zipHypo.count() )
                //      {
                //          FILE* logFile = fopen( "wapp.log", "a+" );
                //
                //          if( logFile )
                //          {
                //              char buf[ UTT_PROMPT_LENGTH ] = "";
                //
                //              if( t )
                //                  zipUtt.resultName( buf, sizeof( buf ), i );
                //
                //              fprintf( logfile, "ResultWord = %ld = %s", (unsi
 => gned long)t, buf );
                //
                //              fclose( logfile );
                //          }
                //
                //          assert( j != zipHypo.count() );
                //      }

        }
```

```
        zipHypo.removeAll();

        if( k )
                zipHypo.makeRoom( k );

        for( i=0; i<k; ++i )
        {
                new( (void* )&zipHypo[ i ] )  ZipHypothesis( test[ i ] );

                // order matters, it is important we fill the cityHypo in the or
 => der
                // of the most probable zipHypothesis to the least probable...
                StateZip::zipToStateCities( &zipHypo[ i ], &cityHypo );

                //      assert( zipHypo[ i ].hasIntegrity() );
        }
}


/////////////////////////////////////////////////////////////////////////
//  Function called in onSpeech.
//
// When recognized city name does not match zip hypotheses
// We are going to collect the zips for the city that should go with this
// city, but that we somehow never hypothesized. If that new zip scores
// well, take it.
//
/////////////////////////////////////////////////////////////////////////


CityHypothesis* DigWin::rerecogZip( int* foundDistance )
{
        // Number of words on choice list. The utt channel knows the  choice lis
 => t
        int cnt = cityUtt.resultCount();

        if( cnt == 0 )
                return 0;

        static SD_STATE tmpState=0;

        // Create and set new ZIP state
        tmpState = recog->newState( "tmpzip" );

        recog->setState( tmpState );

        // Get the range of zips for this state
        StateZip* sz = StateZip::findStateZip( zipResult.stateName() );

        ZipHypothesis* zh = 0;

        // if StateZip is NOT NULL
        if( sz )
        {
                // Do not use more than 3 top candidates of city choice list
                if( cnt > 3 ) cnt = 3;

                // Clean up city hypotheses and zip hypotheses
                cityHypo.removeAll();
                zipHypo.removeAll(); // of type ZipHypoAC (array)

                zipHypo.makeRoom( 1 );
```

```
		// initialize new choice list for zips
		new( &(zipHypo[0]) )  ZipHypothesis( recog, zipResult );

		// cnt  == 3
		for( int icity=0; icity < cnt; ++icity )
		{
			// Get icity-th choice from city choice list
			cityUtt.resultName( buf, sizeof(buf), icity );

			// watch out for canceladdress...
			// FIX LATER

			CityHypothesis* ch = &( cityHypo[ icity ] );

			zips.removeAll();

			// Get zips for city name, and return city name
			ch->cityName = az->cityToZips( buf, &zips );
			ch->zh = 0;
			ch->az = az;
			ch->zipAgreesWithState = 1;

			int imax = zips.count();

			// build new choice list for the zips found for the 3 ci
=> ty top
			// choices
			if( imax )
			{
				int i0 = zipHypo.makeRoom( imax );

				for( int i=0; i < imax; ++i )
				{
					// Build a new zip choice list zipHypo (
=> of type ZipHypoAC)
					zh = ( new( &(zipHypo[ i0 + i ]) )
								ZipHypothesis(
=> recog, zips[ i ], zipResult ) );

					// zh is zipHypothesis; ch == cityHypoth
=> esis
					zh->ch = ch;

					//assert( zh->hasIntegrity() );
				}
			} // if imax
		} // for ( cnt  == 3)

		// zipHypothesis
		zh = 0;

		UttChannel *chan = recog->getChannel();

		recog->setChannel( &zipUtt );

		// Discrete recognition in in tmpState words in there
		// the words in this state are the 5 digit built phrases.
		if( recog->recog() == 0 )
		{
			// Get the number of choices
			cnt= zipUtt.resultCount();

			if( cnt > 1 )
			{
				SD_WORD w = zipUtt.resultId();
```

```
				// there is a match of SD_WORD id's
				int i = (w == zipHypo[0].wordId);

				w = zipUtt.resultId( i );

				*foundDistance = zipUtt.resultDistance( i );

				for( i=0; i<cnt; ++i )
				{
					if( zipHypo[ i ].wordId == w )
					{
						// get zipHypothesis from zipHyp
=> o
						zh = &(zipHypo[ i ]);

						zh->ch->zh = zh;

						break;
					}
				} // for
			} // if( cnt == 1)
		}

		// clean up
		zipHypo.cleanUpNewWords( recog );

		recog->setChannel( chan );
	}

	// clean up temp state
	recog->deleteState( tmpState );

	return( zh != 0 ? zh->ch : 0 );
}


// Helper function

bool DigWin::isCertain( SD_WORD testWord )
{
	recog->setVoc( cityVoc );

	if( testWord == 0 )
		return 0;

	recog->setState( (SD_STATE )0 );		// open Recognition

	if( 0 == recog->recog() )
	{
		SD_WORD wordToCompareWith = recog->resultId();

		if( wordToCompareWith != testWord && wordToCompareWith != 0 )
		{
			SD_STATE st = recog->setState( testState );

			//		if( st == 0 )
			//			return 0;

			assert( st != 0 );

			recog->addWord( wordToCompareWith );
			recog->addWord( testWord );

			recog->recog();
```

```
                recog->deleteWord( wordToCompareWith );
                recog->deleteWord( testWord );
        }

        if( recog->resultId() == testWord && recog->confidence() > 85 )
        {
                if( testWord == newfoundlandSL.wordId )
                {
                        recog->setState( "digits" );

                        recog->addWord( testWord );

                        recog->recog();

                        recog->deleteWord( testWord );

                        int dist = recog->resultDistance();

                        if( dist != -1 )
                                return 0;
                }

                recog->resultName( buf, sizeof( buf ) );
                recog->setPrompt( buf );
                recog->adapt();

                return 1;
        }
    }

    return 0;
}

//////////////////////////////////////////////////////////////////////////
//
//
//
//////////////////////////////////////////////////////////////////////////

bool DlgWin::isCancelBadLabelGoToSleep( HWND hwnd )
{
        CustomCitySL test( 0, "", recog->resultId(), 0 );

        CustomCitySL* found = zipCommandWordsHS.find( &test );

        if( found != 0 && found->isControlWord && isCertain( test.wordId ) )
        {
                // wrong answer, reset to ZIP_STATE
                if( test.wordId == cancelWord )
                {
                        recog->beep( "bad.wav" );

                        appState = ZIP_STATE;
                }

                // zip and city do not make sense, reset to ZIP_STATE
                else if( test.wordId == badLabelWord )
                {
                        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), "Bad
```

WAPP.CPP 7-22-95 11:36a

```
=> Label" );
                        numErrors++;

                        outCommPort( ltoZip( badLabelSL.zip ), "BadLabel", "BadL
=> abel" );

                        appState = ZIP_STATE;
                }

                // correct
                else if( test.wordId == sameZipWord )
                {
                        outCommPort( lastZipOutCommPort, lastStateOutCommPort, l
=> astCityOutCommPort );
                        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), last
=> ZipOutCommPort );
                        SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_ID ), la
=> stStateOutCommPort );
                        SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_ID ), las
=> tCityOutCommPort );

                        recog->killUtt();

                        return 1;
                }

                // Command word: Go to Sleep
                else if( test.wordId == goToSleepWord )
                {
                        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), "Sle
=> eping" );

                        recog->beep( "sleeping.wav" );

                        appState = WAKEUP_STATE;
                }

                // Fall through
                else if( test.wordId != wakeUpWord )
                        return 0;

        }// if (found)

        // Nothing found
        else
                return 0;

        SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_ID ), "" );

        SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_ID ), "" );

        // Clean up
        recog->killUtt();

        return 1;
}

//////////////////////////////////////////////////////////////////////////
```

Page 15 of 29

```
//
//
//
////////////////////////////////////////////////////////////////////////////

bool DlgWin::isCancelBadLabelGoToSleepOrProvince( HWND hwnd )
{
        CustomCitySL test( 0, "", recog->resultId(), 0 );

        CustomCitySL* found = zipCommandWordsHS.find( &test );

        if( found != 0 && isCertain( test,wordId ) )
        {
                if( found->isControlWord )
                {
                        // Command word == CANCEL
                        if( test.wordId == cancelWord )
                        {
                                recog->beep( "bad.wav" );

                                appState = ZIP_STATE;
                        }


                        // Command word == answer makes no sense
                        // When a package keeps on being rejected, the
                        // person says "Bad label", which will send to package
                        // to a special place for address inspection
                        else if( test.wordId == badLabelWord )
                        {
                                SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID
  =>  ), "BadLabel" );

                                numErrors++;

                                outCommPort( ltoZip( badLabelSL.zip ), "BadLabel
  => ", "BadLabel" );

                                appState = ZIP_STATE;
                        }

                        // Correct
                        else if( test.wordId == sameZipWord )
                        {
                                // send out answer
                                outCommPort( lastZipOutCommPort, lastStateOutCom
  => mPort, lastCityOutCommPort );

                                SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID
  =>  ), lastZipOutCommPort );
                                SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_
  => ID ), lastStateOutCommPort );
                                SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_I
  => D ), lastCityOutCommPort );

                                recog->killUtt();

                                return 1;
                        }

                        // Command word == Go to Sleep
                        else if( test.wordId == goToSleepWord )
                        {
                                SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID
  =>  ), "Sleeping" );
```

WAPP.CPP 7-22-95 11:36a

```
                                recog->beep( "sleeping.wav" );

                                appState = WAKEUP_STATE;
                        }

                        // Fall through
                        else if( test.wordId != wakeUpWord )
                                return 0;
                } // if( found->isControlWord )

                else
                {
                        // its a province...    ship it...
                        // No further info about cities is needed.
                        outCommPort( ltoZip( found->zip ), found->name, "Pure" )
  => ;

                        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), foun
  => d->name );
                }
        } // if( found != 0 && isCertain( test.wordId ) )

        else
                return 0;

        SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_ID ), "" );

        SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_ID ), "" );

        recog->killUtt();

        return 1;
}


// Helper function

void DlgWin::itsBad( HWND hwnd, UttChannel* toMe )
{
        // the city recognition ran into a problem
        if( toMe == &cityUtt  && !fixError )
        {
                numErrors++;

                outCommPort( ltoZip( badLabelSL.zip ), "BadLabel", "BadLabel" );

                SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), "BadLabel" )
  => ;
        }

        // the zip state recognition failed.
        else
        {
                recog->beep( "bad.wav" );

                SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), "no recognit
  => ion" );

                //Stijn suggests: bad counter can be set and brought to screen h
  => ere.
        }

        SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_ID ), "" );
```

Page 16 of 29

```
        SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_ID ), "" );

        recog->giveAwayUtt( toMe );

        toMe->setPrompt( "" );

        appState = ZIP_STATE;
}


void DlgWin::itsGoodAdapt( CityHypothesis& ch )
{
        ZipHypothesis* zh = ch.zh;

        //    assert( zh->hasIntegrity() );

        choiceListStats.recordResult( zh->zipStateName() );

        if( zh->zipStateName() != zh->spelling )
        {

                char *s = ( zh->spelling[ 0 ] == 'b'
                                                                  ? bluePrefix
                                                                  : ( zh->spelling
  => [ 0 ] == 'o'

  => ? orangePrefix

  => : 0 ) );

                assert( s != 0 );

                strncpy( s + 2, zh->zipStateName(), 3 );

                s[ 5 ] = '\0';

                outCommPort( s, ch.sz->stateName(), ch.cityName );
        }

        else
                outCommPort( zh->spelling, ch.sz->stateName(), ch.cityName );

        // lets adapt...
        recog->setVoc( cityVoc );
        recog->setChannel( &cityUtt );
        recog->setPrompt( ch.cityName );
        recog->adapt();

        if( ch.zipAgreesWithState )
        {
                recog->setVoc( digitsVoc );
                recog->setChannel( &zipUtt );
                recog->setPrompt( zh->phrase );
                recog->contAdapt();
        }

        recog->setChannel( channel );

        *(zh->zipStateName() + 5) = '\0';

        SetWindowText( GetDlgItem( hWnd, REC_ZIP_TEXT_ID ), zh->spelling );
        SetWindowText( GetDlgItem( hWnd, REC_STATE_TEXT_ID ), ch.sz->stateName()
  =>  );
        SetWindowText( GetDlgItem( hWnd, REC_CITY_TEXT_ID ), ch.cityName );
```

```
        appState = ZIP_STATE;

        static int count = 0;
        if( count++ == 4 )
        {
                count = 0;
                static writeCount = 0;
                static long write[ 4 ];

                char num[ 20 ];
                ltoa( write[ writeCount ]=computeMemoryUse(), num, 10 );
                SetWindowText( GetDlgItem( hWnd, MEMORY_TEXT_ID ), num );

                if( ++writeCount == 4 )
                {
                        writeCount = 0;
                        FILE * fout = fopen( "memuse.log", "a+" );
                        if( fout )
                        {
                                fprintf( fout, "%ld\n", write[ 0 ] );
                                fprintf( fout, "%ld\n", write[ 1 ] );
                                fprintf( fout, "%ld\n", write[ 2 ] );
                                fprintf( fout, "%ld\n", write[ 3 ] );
                                fclose( fout );
                        }
                }
        }
}


////////////////////////////////////////////////////////////////////////////
//
//      onSpeech is the heart of the solution.
//
////////////////////////////////////////////////////////////////////////////


void DlgWin::onSpeech( HWND hwnd, SD_CHANNEL ch )
{
        // get the utterance handle; do not wait for the end of the
        // utterance since the recognition can proceed in parallel
        // with the utterance collection

        assert( recog );

        // notify the dragcpp classes for speech event
        if( recog->notifyChannel( ch, SD_CHANNEL_START ) )
        {

                // reset list box content
                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_RESETCONTENT, 0, 0
  => );


                // appState is initialized to ZIP_STATE
                switch( appState )
                {
                 case ZIP_STATE:
                        {
                                // reset variables to NULL and
                                // remove hypotheses from previous looping

                                zipResult.spelling[ 0 ] = '\0';
                                zipResult.phrase[ 0 ] = '\0';
```

```
                                   digitSpelling[ 0 ] = '\0';

                                   int wasGoodZipPhrase = 0;

                                   zipHypo.removeAll();

                                   SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID
=>  ), "" );
                                   SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_
=> ID ), "" );
                                   SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_I
=> D ), "" );

                                   // Get digit vocabulary
                                   recog->setVoc( digitsVoc );
                                   // Get digit0color state
                                   recog->setState( "digit0color" );

                                   // start recognition
                                   if( 0 == recog->contRecog() )
                                   {
                                           /*      .........................
=> ..........................
                                           Stage 1: Recognition of zip code and sta
=> te name.
                                           Get the result and analyse its plausibli
=> ty.

                                           This the format the recognizer returns:
                                           "blue 0 1 9 1 3 massachusetts"

                                           This answer is stored as a phraseSpellin
=> g, which is turned into a
                                           zipStateSpelling. zipStateSpelling does
=> not have spaces between the
                                           digits.

                                           phraseSpelling          "blue 0 1 9 1 3
=> massachusetts"
                                           zipStateSpelling        "01913 massachus
=> etts"
                                           stateSpelling           "massachusetts"
                                           digitSpelling           "01913"

                                           The class recog has knowledge about the
=> class channel. The channel
                                           knows the acoustics and the word label t
=> hat goes with the acoustics.
                                           When the channel is a file ( == script)
=> , then the word comes
                                           from a list file. When the channel is li
=> ve speech, the label
                                           is what the recognizer thinks it is.

                                           The function recog->resultname() looks i
=> nto the channel for the
                                           label that goes with the acoustics, and
=> copies it into
                                           zipResult.phrase. This function has a 3r
=> d parameter which refers
                                           to the ith item on the recognition list
=> returned by recog(). However
                                           for continuous speech there is only one
                         WAPP.CPP  7-22-95   11:36a
```

```
=> item on the list.                        (The discrete recognizer can ret
=> urn many more.)
                                            ......................................
=> ....................*/

                                            recog->resultName( zipResult.phrase, siz
=> eof( zipResult.phrase ) );

                                            /* ...................................
=> ........................

                                            zipResults.initFromPhrase returns number
=>  of recognized words.
                                            A good recognition returns 5 or 6 words.
                                            The correct number is 6: 5 digits and a
=> state name.     Sometimes
                                            5 words were returned. We will try to re
=> cover from this error
                                            in the function buildHypothesis().

                                            Kinds of misrecognition:
                                            1. a deletion error in the digits, ..88.
=>  -> ..8.
                                            2. state name recognized as 3 or 8.

                                            If wasGoodZipPhrase, of type integer, is
=>  less than 5,
                                            we consider it not worth pursuing.

                                            ......................................
=> ........................*/

                                            wasGoodZipPhrase = zipResult.initFromPhr
=> ase();

                                            // Copy digits, not the state name
                                            strncpy( digitSpelling, zipResult.zipSta
=> teName(), 5 );

                                            digitSpelling[ 5 ] = '\0';

                                            // copy digit string into ChoiceListStat
=> s class
                                            choiceListStats.setOpenRecResult( digitS
=> pelling );

                                    }

                                    /*.........................................
=> .........

                                    Stage 2: re-recognition stage against single
                                    word commands...

                                    ..........................................
=> ..........*/

                                    // A: PREPARATION:

                                    Page 18 of 29
```

```
                                       // Set voc file and state. Get Word ID's of comm
=> on
                                       // commands and provinces, which are dealt with
=> differently,
                                       // since they do not have a city recognition sta
=> ge.

                                       // Open the voc with all the city names
                                       recog->setVoc( cityVoc );
                                       // Go to the state with all the command words( g
=> otosleep, cnacel etc...)
                                       recog->setState( "zipcommandwords" );

                                       // cancelWord is of type SD_WORD. It was initial
=> ized to 0.
                                       if( cancelWord == 0 )
                                       {
                                               // Is used somewhere in function isCerta
=> in
                                               testState = recog->getState( "testCertai
=> ntyState" );

                                               /*........................................
=> ..................
                                               Get word ID's (type SD_WORD) of importan
=> t commands, such as Wake Up,
                                               and special states, such as alberta, we
=> will need all the time.
                                               These ID's are global.
                                               ..........................................
=> ..................*/

                                               cancelWord = cancelAddressSL.wordId = re
=> cog->wordId( "canceladdress" );
                                               zipCommandWordsHS.add( &cancelAddressSL
=> );

                                               badLabelWord = badLabelSL.wordId = recog
=> ->wordId( "badlabel" );
                                               zipCommandWordsHS.add( &badLabelSL );
                                               goToSleepWord = goToSleepSL.wordId = rec
=> og->wordId( "gotosleep" );
                                               zipCommandWordsHS.add( &goToSleepSL );
                                               wakeUpWord = wakeUpSL.wordId = recog->wo
=> rdId( "wakeup" );
                                               zipCommandWordsHS.add( &wakeUpSL );
                                               sameZipWord = sameZipSL.wordId = recog->
=> wordId( "samezip" );
                                               zipCommandWordsHS.add( &sameZipSL );
                                               mexicoSL.wordId = recog->wordId( "mexico
=> " );
                                               zipCommandWordsHS.add( &mexicoSL );
                                               albertaSL.wordId = recog->wordId( "alber
=> ta" );
                                               zipCommandWordsHS.add( &albertaSL );
                                               britishColumbiaSL.wordId = recog->wordId
=> ( "britishcolumbia" );
                                               zipCommandWordsHS.add( &britishColumbiaS
                           WAPP.CPP  7-22-95  11:36a
```

```
=> L );
                                               labradorSL.wordId = recog->wordId( "labr
=> ador" );
                                               zipCommandWordsHS.add( &labradorSL );
                                               manitobaSL.wordId = recog->wordId( "mani
=> toba" );
                                               zipCommandWordsHS.add( &manitobaSL );
                                               newBrunswickSL.wordId = recog->wordId( "
=> newbrunswick" );
                                               zipCommandWordsHS.add( &newBrunswickSL )
=> ;

                                               newFoundlandSL.wordId = recog->wordId( "
=> newfoundland" );
                                               zipCommandWordsHS.add( &newFoundlandSL )
=> ;

                                               northWestTerritoriesSL.wordId = recog->w
=> ordId( "northwestterritories" );
                                               zipCommandWordsHS.add( &northWestTerrito
=> riesSL );

                                               novaScotiaSL.wordId = recog->wordId( "no
=> vascotia" );
                                               zipCommandWordsHS.add( &novaScotiaSL );
                                               ontarioSL.wordId = recog->wordId( "ontar
=> io" );
                                               zipCommandWordsHS.add( &ontarioSL );
                                               princeEdwardIslandSL.wordId = recog->wor
=> dId( "princeedwardisland" );
                                               zipCommandWordsHS.add( &princeEdwardIsla
=> ndSL );

                                               quebecSL.wordId = recog->wordId( "quebec
=> " );
                                               zipCommandWordsHS.add( &quebecSL );
                                               saskatchewanSL.wordId = recog->wordId( "
=> saskatchewan" );
                                               zipCommandWordsHS.add( &saskatchewanSL )
=> ;

                                               yukonTerritorySL.wordId = recog->wordId(
=>  "yukonterritory" );
                                               zipCommandWordsHS.add( &yukonTerritorySL
=>  );
                                       }

                                       //..........................................
=> ...........
                                       // 8: Build ContRecog Choice list.

                                       // The contRecog only returns
                                       // one choice. We will generate some alternative
=> s based on
                                       // a confusability matrix, designed for the digi
=> ts.

                                       // Value of wasGoodZipPhrase needs to be 5 or 6.
                                       // It is probably always true.
                                       if( wasGoodZipPhrase )
                                        Page 19 of 29
                                        // !!!!!!!!!    Important
                                        // Build zip code hypotheses, using stac
=> k decoder. Look at
                                        // buildHypothesis. The parameter wasGoo
=> dZipPhrase is passed
                                        // so that we know whether to apply an e
=> rror correction
                                        // technique to make an educated guess f
=> or 6 likely words.

                                        // As a result of this function we will
=> have a choice list of
                                        // phrases containing the zip hypothesis
=>  and state name. Each
                                        // phrase, there are 40 of them, has its
=>  own SD_WORD id added to
                                        // the current state zipCommandState The
=>  &zipHypo class is now
                                        // filled with 40 zips and the state nam
=> e.

                                        // The return value is 6 if an error cor
=> rection could be made, else
                                        // 5 is returned.
                                        //-------------------------------------
=> -------------------

                                        wasGoodZipPhrase = buildHypothesis( reco
=> g, &zipHypo, zipResult, wasGoodZipPhrase );

                                  //-------------------------------------------
=> -----------
                                  // C: RE-RECOGNIZE against phrases.
                                  // Discrete recognition is used against the phra
=> ses stored in
                                  // zipHypo to get the best recognition results a
=> gainst the utterance
                                  // in the speech queue. A choice list is returne
=> d, standard feature
                                  // for discrete recognition.
                                  // We are still in commandWordState for recognit
=> ion
                                  //----------------------------------   -------
=> -----------

                                  if( recog->recog() == 0 )
                                  {
                                          // i < number of words on choice list
                                          for( int i = 0; i < recog->resultCount()
=> ; ++i )
                                          {
                                                  recog->resultName( buf, sizeof(
=> buf ), i );

                                                  SendDlgItemMessage( hwnd, CHOICE
=> _LIST_ID, LB_ADDSTRING, 0, (DWORD )buf );
                                          }

                                          // clean up the new words added by phras
=> e building from
```

```
                                        // the zipCommandState (and Voc) (delete
=>  them)
                                        // zipHypo still knows about the choices
                                        zipHypo.cleanUpNewWords( recog );

                                        // class recognizer knows about the retu
=> rned choice list, built up
                                        // in decreasing confidence order.
                                        addressConfidence = recog->confidence();

                                        // If the confidence is lower than 4, we
=>  do not trust our
                                        // answer. The recognizer tells us that
=> it is not very certain
                                        // about what it found. We will reject t
=> he answer.
                                        if( addressConfidence < 4 )
                                        {
                                                itsBad( hwnd, &zipUtt );
                                                break;
                                        }

                                        // It is a command word or a special sta
=> te like alberta...
                                        // Remember we are in the commandState
                                        else if( isCancelBadLabelGoToSleepOrProv
=> ince( hwnd ) )
                                        {
                                                break;
                                        }

                                        // bingo: display our best guess for zip
=>  and state
                                        else if( wasGoodZipPhrase == 6 )
                                        {
                                                SetWindowText( GetDlgItem( hwnd,
=>  REC_ZIP_TEXT_ID ), digitSpelling );
                                                SetWindowText( GetDlgItem( hwnd,
=>  REC_STATE_TEXT_ID ), zipResult.stateName() );

                                                // Move this utt into the zipUtt
=>  channel If not moved or killed, the
                                                // channel will keep giving the
=> same utt back.
                                                recog->giveAwayUtt( &zipUtt );

                                                // ----------------------------
=> ------------------------

                                                // D: BUILD MORE HYPO's,
                                                // Use the zip and state info to
=>  make guesses about the city we
                                                // might expect to be said.
                                                // We do this now since we could
=>  have some spare time here
                                                // to do this computation, inste
=> ad of later, when we are in
                                                // the CITY_STATE. In the CITY_S
=> TATE we will continue working
                                                // with this information.
                                                // ----------------------------
=> ------------------------

                                                buildCityHypothesis();

                                                // get number of city hypotheses
                                                if( cityHypo.count() )
```

```
                                               {
                                                      CityHypothesis* ch = &(c
=> ityHypo[ 0 ]);
                                                      ZipHypothesis* zh = ch->
=> zh;
                                                      // assert( zh->hasIntegr
=> ity() );
                                                      if( ch->zipAgreesWithSta
=> te && ch->sz->isPure() )
                                                      {
                                                              outCommPort( zh-
=> >spelling, zh->stateName(), "pureState"  );
                                                              appState = ZIP_S
=> TATE;
                                                              break;
                                                      }
                                                      // set recognizer to voc
=> al: state with name of the
                                                      // recognized state to g
=> et the SD_WORD id's for each city
                                                      // and put them in the c
=> ityHypo class.

                                                      // later this info will
=> be moved into a
                                                      // tmpState. can't we do
=>  this directly? Take top
                                                      // 6 zips and get the ci
=> ties with those zips
                                                      recog->setState( zipResu
=> lt.stateName() );

                                                      for( i = cityHypo.count(
=> ); i--; )
                                                      {
                                                              cityHypo[ i ].wo
=> rdId = recog->wordId( cityHypo[ i ].cityName );
                                                      }
                                                      // We have a number of a
=> nswers for the zip and state
                                                      // set appState to CITY_
=> STATE, to do recognition on the cities
                                                      appState = CITY_STATE;

                                                      break;
                                               }
                                       }
                                       itsBad( hwnd, &zipUtt );

                                       break;
                               }
                               // else recog failed!!!

                               // clean up
                               zipHypo.cleanUpNewWords( recog );

                               itsBad( hwnd, &zipUtt );
                          WAPP.CPP  7-22-95  11:36a
```

```
                      } // case ZIP_STATE
                      break;

                      //------------------------------------------------------
=> -------
                      // Now we are out of the message loop. Our app is waitin
=> g
                      // for a new message from the FEP for speech. The next t
=> ime
                      // speech is heard we will be in the state CITY_STATE.

                      // The next step will have 2 discrete recognition steps.
               case CITY_STATE:
                      {
                              // First discrete recognition step: The state we
=>  recognize
                              // against is the "StateName" state, set at the
=> end of
                              // ZIP_STATE.

                              int i = ( 0 == recog->recog() );

                              // Create a new state
                              SD_STATE tmpState = recog->newState( "tmpcity" )
=> ;

                              // Point to tmpCity state
                              recog->setState( tmpState );

                              // DebugOutput( DBF_ERROR, "Just a Test" );

                              // recognition worked (i = ( 0 == recog->recog()
=> )
                              // Put the choice list of cities in tmpState
                              if( i )
                              {
                                      for( i = 0; i < recog->resultCount(); ++
=> i )
                                      {
                                              recog->addWord( recog->resultId(
=>  i ) );
                                      }
                              }

                              // We already built some hypo's in the ZIP_STATE
=>  above. Now we will
                              // be adding the SD_WORD id's to the tmpState.
                              for( i = cityHypo.count(); i--; )
                              {
                                      SD_WORD w = cityHypo[ i ].wordId;

                                      if( w != 0 )
                                              // add the city ID's from cityHy
=> po to the tmpState
                                              recog->addWord( w );
                              }


                              // Second discrete recognition on city utt
                              // We do another discrete recognition, but now a
=> gainst
                              // the tmpState, which is filled up with city na
=> mes
                              // from logical guesses in ZIP_STATE, and with c
=> ity names
                               Page 21 of 29
```

```
                                    // found by discrete recognition against the "st
=> ate" STATE
                                    // in the city VOC.

                                    if( 0 == recog->recog() )
                                    {
                                            // cleanup first...
                                            recog->deleteState( tmpState );

                                            addressConfidence = recog->confidence();

                                            //                      for( i=0; i<zipHypo.
=> count(); ++i )
                                            //                              zipHypo[ i ].has
=> Integrity();


                                            // Put the choice list of cities on scre
=> en
                                            for( i = 0; i < recog->resultCount(); ++
=> i )
                                            {
                                                    // get city name
                                                    recog->resultName( buf, sizeof(
=> buf ), i );

                                                    SendDlgItemMessage( hwnd, CHOICE
=> _LIST_ID, LB_ADDSTRING, 0, (DWORD )buf );
                                            }
                                            // Move the utt in the channel to cityUt
=> t
                                            // Move the utt now, before we check for
=>  isCancel...
                                            // otherwise we blow away the choice lis
=> t of
                                            // the tmpState recognition
                                            recog->giveAwayUtt( &cityUtt );

                                            // if it is a command word, perform comm
=> and
                                            if( isCancelBadLabelGoToSleep( hwnd ) )
                                            {
                                                    break;
                                            }


                                            // declare new array of SD_WORD
                                            static AC< SD_WORD > cityResultIds;

                                            // Clean up previous ID's
                                            cityResultIds.removeAll();

                                            // no more then 20 are kept around
                                            int cnt = cityUtt.resultCount();

                                            for( i=0; i < cnt; ++i )
                                                    cityResultIds.add( cityUtt.resul
=> tId( i ) );

                                            for( i=0; i<cnt; ++i )
                                            {
                                                    for( int j=cityHypo.count(); j--
=> ; )
                                                    {
                                                            // A good answer
```

WAPP.CPP 7-22-95 11:36a

```
                                                                    if( cityHypo[ j ].wordId
=>  == cityResultIds[ i ] )
                                                                    {
                                                                            // zip matches a
=> ome city
                                                                            itsGoodAdapt( ci
=> tyHypo[ j ] );

                                                                            // The end of th
=> e control flow
                                                                            goto EndOfSwitch
=> ;
                                                                    }
                                                            }
                                                    }
                                                    // if we got here things didn't match up
=> ...
                                                    // try rerecognizing zip with the best c
=> ity's zips...

                                                    int foundDistance;
                                                    CityHypothesis* ch;

                                                    // re-recognize: This function is anothe
=> r heavy duty one
                                                    // Please look there as well,
                                                    // returns city hypothesis which knows a
=> bout the zipHypothesis
                                                    if( 0 != (ch=rerecogZip( &foundDistance
=> )) )
                                                    {
                                                            addressConfidence = 100 - foundD
=> istance;

                                                            // the first choice does not mak
=> e sense
                                                            if( addressConfidence < 0 )
                                                                    addressConfidence = 0;

                                                            // the confidence is good.
                                                            // In this case, the city was
                                                            if( addressConfidence > 0 )
                                                            {
=>  zip state city
                                                                    // everything matches...

                                                                    itsGoodAdapt( *ch );

                                                                    break;
                                                            }
                                                    }

                                                    // else its Wrong...
                                                    itsBad( hwnd, &cityUtt );

                                                    break;
                                            }
                                            // cleanup temp state that contains all the educ
=> ated guesses
                                            // about city recognition
                                            recog->deleteState( tmpState );

                                            // no recognition...
                                            itsBad( hwnd, &cityUtt );

                                            break;
                                    }
```

Page 22 of 29

```
                    // We are sleeping, and expecting to wake up.
                    // Re initialize text on screen.

                case WAKEUP_STATE:
                    {
                        if( isCertain( wakeUpWord ) )
                        {
                            SetWindowText( GetDlgItem( hwnd, REC_ZIP
  => _TEXT_ID ), "Awake" );

                            recog->beep( "awake.wav" );

                            SetWindowText( GetDlgItem( hwnd, REC_STA
  => TE_TEXT_ID ), "" );

                            SetWindowText( GetDlgItem( hwnd, REC_CIT
  => Y_TEXT_ID ), "" );

                            appState = ZIP_STATE;
                        }

                        recog->killUtt();
                    }
                    break;
                }
EndOfSwitch:
                // we always get to this end switch
                SetWindowText( GetDlgItem( hwnd, CONFIDENCE_TEXT_ID ), itoa( add
  => ressConfidence, buf, 10 ) );

                SetWindowText( GetDlgItem( hwnd, PROMPT_TEXT_ID ), sayPrompt[ ap
  => pState ] );

                // In case there is still another utterance, but we have no star
  => t of
                // utterance message, put a start message in the queue.
                if( recog->peekUtt() )
                {
                    postSpeechEvent( (SD_CHANNEL )1, SD_CHANNEL_START );
                }
        }
}


void DlgWin::onSysCommand( HWND hwnd, UINT cmd, int, int )
{
        switch( cmd )
        {
         case SC_CLOSE:
                EndDialog( hwnd, 0 );
                break;
        }
}

// Function that organizes the user interface

void DlgWin::onCommand( HWND hwnd, UINT cmd, HWND, UINT )
{
        switch ( cmd )
        {
         case CHOICE_LIST_ID:
                break;

         case USR_COMBO_ID:
```

WAPP.CPP 7-22-95 11:36a

```
                break;

        case SAVE_BTN_ID:
                if( recog && channel && recog->getUser() )
                {
                        channel->flush();

                        recog->saveUser( origCityUser );
                        cityUser = origCityUser;

                }
                break;

        case EXIT_BTN_ID:
                EndDialog( hwnd, 0 );
                break;

        case FIX_ERRORS_BTN_ID:

                fixError = !fixError;

                SetWindowText( GetDlgItem( hwnd, FIX_ERRORS_BTN_ID ),
                                                                      fixErro
=> r ? "Fix &Errors" : "Ignore &Errors" );
                break;


        case PURGE_BTN_ID:
                if( speechTask )
                {
                        HCURSOR hCursor = SetCursor( LoadCursor( 0, IDC_WAIT ) )
=> ;

                        zipUtt.killUtt();
                        cityUtt.killUtt();

                        int hadListened =  channel != 0 && recog != 0;

                        int wasListening = hadListened && channel->isListening()
=> ;

                        if( wasListening )
                                channel->listen( FALSE );

                        if( hadListened )
                        {
                                recog->saveUser( tmpCityUsr );
                                cityUser = tmpCityUsr;
                        }

                        if( channel )
                        {
                                delete channel;
                                channel = 0;
                        }

                        if( recog )
                        {
                                delete recog;
                                recog = 0;
                        }
                        delete speechTask;

                        speechTask = new SpeechTask( name, reportError );
```

Page 23 of 29

```
                        if( hadListened )
                        {
                                channel = new WindowsLiveChannel();

                                recog = new Recognizer( channel );

                                if(
                                        recog->setUser( cityUser ) == 0 ||
                                        recog->setVoc( digitsVoc ) == 0 ||
                                        recog->setVoc( cityVoc ) == 0 )
                                {
                                        MessageBox( hwnd, "Could not open either
=>  digits.voc or ctall.voc or user file!",
=>          DlgWin::name, MB_ICONEXCLAMATION | MB_OK );

                                        EndDialog( hwnd, 0 );
                                }

                                // handle the callback function to the FEP
                                // Look in dragcpp.cpp for more details how the
=> recognizer
                                // class deals with all this
                                channel->open( postSpeechEvent );

                                if( wasListening )
                                        channel->listen( TRUE );
                        }

                        SetCursor( hCursor );
                }

                break;


        case SETUSER_BTN_ID:
                {
                        char usrName[ UTT_PROMPT_LENGTH ];

                        if( !GetDlgItemText( hwnd, USR_COMBO_ID, usrName, sizeof
=> (usrName) ) )
                                break;

                        HCURSOR hCursor = SetCursor( LoadCursor( 0, IDC_WAIT ) )
=> ;

                        bool isMale = tolower( usrName[ strlen( usrName ) - 1 ]
=> ) == 'm';

                        if( recog )
                        {
                                extern int fileNameCmp( const char *, const char
=>  * );

                                char buffer[ 128 ];
                                recog->getUsername( buffer, sizeof( buffer ) );

                                if( fileNameCmp( usrName, buffer ) )
                                {
                                        bool wasListening = channel->isListening
=> ();

                                        if( wasListening )
                                                channel->listen( FALSE );

                WAPP.CPP  7-22-95  11:36a
```

```
                                        recog->closeVoc( cityVoc );
                                        recog->closeVoc( digitsVoc );
                                        recog->closeUser( cityUser );

                                        cityUser = origCityUser;

                                        strcpy( cityUser, usrName );

                                        if( isMale )
                                        {
                                                digitsVoc = malDigitsVoc;
                                                cityVoc = malCityVoc;
                                        }
                                        else
                                        {
                                                digitsVoc = femDigitsVoc;
                                                cityVoc = femCityVoc;
                                        }

                                        if(
                                                 recog->setUser( cityUser ) == 0
=>  ||
                                                 recog->setVoc( digitsVoc ) == 0
=>  ||
                                                 recog->setVoc( cityVoc ) == 0 )
                                        {
                                                MessageBox( hwnd, "Could not ope
=> n either digits.voc or ctall.voc or user file!",
=>                      DlgWin::name, MB_ICONEXCLAMATION | MB_OK );

                                                EndDialog( hwnd, 0 );
                                        }

                                        if( wasListening )
                                                channel->listen( TRUE );
                                }
                        }
                        else    // recog == 0
                        {
                                // Load male vocabulary
                                if( isMale )
                                {
                                        digitsVoc = malDigitsVoc;
                                        cityVoc = malCityVoc;
                                }

                                // Load female vocabulary
                                else
                                {
                                        digitsVoc = femDigitsVoc;
                                        cityVoc = femCityVoc;
                                }

                                // Get the channel
                                if( channel == 0 )
                                        channel = new WindowsLiveChannel();

                                recog = new Recognizer( channel );

                                strcpy( cityUser, usrName );

                                if(
                                        recog->setUser( cityUser ) == 0 ||
                                        recog->setVoc( digitsVoc ) == 0 ||

                                Page 24 of 29
                                recog->setVoc( cityVoc ) == 0 )
                        {
                                MessageBox( hwnd, "Could not open either
=>  digits.voc or ctall.voc or user file!",
=>          DlgWin::name, MB_ICONEXCLAMATION | MB_OK );
                                EndDialog( hwnd, 0 );
                        }

                        //              FILE *vlistfile = fopen( "vlist"
=> , "w+" );
                        //
                        //              recog->setState( (SD_STATE )0 );
                        //
                        //              recog->listState( 50000, vlistfi
=> le );
                        //
                        //              fclose( vlistfile );

                        channel->open( postSpeechEvent );

                        EnableWindow( GetDlgItem( hwnd, MICROPHONE_BTN_I
=> D ), TRUE );

                        //              EnableWindow( GetDlgItem( hwnd,
=> SETUSER_BTN_ID ), FALSE );

                        SetFocus( GetDlgItem( hwnd, WORD_PAUSE_SLIDE_ID
=> ) );

                        numErrors = 0;


                        PostMessage( hwnd, WM_COMMAND, WORD_PAUSE_SLIDE_
=> ID, 0 );
                        PostMessage( hwnd, WM_COMMAND, START_THRESH_SLID
=> E_ID, 0 );
                        PostMessage( hwnd, WM_COMMAND, END_THRESH_SLIDE_
=> ID, 0 );
                }

                SetCursor( hCursor );
            }
            break;

        case MICROPHONE_BTN_ID:
            if( recog )
            {
                char num[ 20 ];
                ltoa( computeMemoryUse(), num, 10 );
                SetWindowText( GetDlgItem( hwnd, MEMORY_TEXT_ID ), num )
=> ;

                recog->listen( !recog->isListening() );

                SetWindowText( GetDlgItem( hwnd, MICROPHONE_BTN_ID ),
=>  recog->isListening() ? "&Microphone On" : "&Microphone Off" );

                SetWindowText( GetDlgItem( hwnd, PROMPT_TEXT_ID ),
=>  recog->isListening() ? sayPrompt[ appState ] : "" );

                if( !recog->isListening() )
                {
                    WAPP.CPP  7-22-95  11:36a
```

```
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_RES
=> ETCONTENT, 0, 0 );

                                long total = right + wrong;
                                if( total == 0 ) total = 1;

                                sprintf( buf, "right:   %ld  %ld\%", right, (10
=> 0*right)/total );
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADD
=> STRING, 0, (DWORD )buf );
                                sprintf( buf, "wrong:   %ld  %ld\%", wrong, (10
=> 0*wrong)/total );
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADD
=> STRING, 0, (DWORD )buf );
                                sprintf( buf, "predicted: %ld  %ld\%", predicted
=> , (100*predicted)/total );
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADD
=> STRING, 0, (DWORD )buf );

                                KillTimer( hwnd, VUTIMER_ID );
                        }
                        else
                        {

                                // initialize vu meter
                                SetTimer( hwnd, VUTIMER_ID, 20, (TIMERPROC )0 );
                        }

                }
                break;


        case WORD_PAUSE_SLIDE_ID:
                {
                        wordPause = GetScrollPos( GetDlgItem( hwnd, WORD_PAUSE_S
=> LIDE_ID ), SB_CTL );

                        bool wasListening = ( recog != 0 && recog->isListening()
=>  );

                        if( wasListening )      recog->listen( 0 );

                        if( speechTask )        speechTask->setPar( "word-pause"
=> , (short )wordPause );

                        if( wasListening )      recog->listen( 1 );

                        char num[ 20 ];

                        itoa( wordPause, num, 10 );

                        SetWindowText( GetDlgItem( hwnd, WORD_PAUSE_TEXT_ID ), n
=> um );
                }
                break;

        case START_THRESH_SLIDE_ID:
                {
                        startThresh = GetScrollPos( GetDlgItem( hwnd, START_THRE
=> SH_SLIDE_ID ), SB_CTL );

                        bool wasListening = ( recog != 0 && recog->isListening()
=>  );

                        if( wasListening )      recog->listen( 0 );

                        if( speechTask )        Fep_Setpar( SUPER_USER, FEP_STAR
                    Page 25 of 29
```

```
=> TOFSPEECHTHRESH,
                                                                        (WORD ) sta
=>
=> rtThresh );
                        if( wasListening )      recog->listen( 1 );

                        char num[ 20 ];

                        itoa( startThresh, num, 10 );

                        SetWindowText( GetDlgItem( hwnd, START_THRESH_TEXT_ID ),
=>  num );
                )
                break;

        case END_THRESH_SLIDE_ID:
                (
                        endThresh = GetScrollPos( GetDlgItem( hwnd, END_THRESH_S
=> LIDE_ID ), SB_CTL );

                        bool wasListening = ( recog != 0 && recog->isListening()
=>  );

                        if( wasListening )      recog->listen( 0 );

                        if( speechTask )        Fep_Setpar( SUPER_USER, FEP_ENDO
=> FOSPEECHTHRESH,
                                                                        (WORD ) end
=>
=> Thresh );

                        if( wasListening )      recog->listen( 1 );

                        char num[ 20 ];

                        itoa( endThresh, num, 10 );

                        SetWindowText( GetDlgItem( hwnd, END_THRESH_TEXT_ID ), n
=> um );
                )
                break;
        )
)

void DlgWin::onTimerDrawVU( HWND hwnd, UINT id )
(
        if( id == VUTIMER_ID && channel && channel->isListening() )
        (
                bool test;

                FEP_S_STATUS fepinfo;

                Fep_Status( (BOOL* )&test, &fepinfo ) ;

                HWND hvu = GetDlgItem( hwnd, VUMETER_ID );

                RECT rc;

                GetClientRect( hvu, &rc );

                //      rc.left      = 95;
                //      rc.right     = 95 + 61;
                //      rc.top       = 116;
                //      rc.bottom    = 116 + 3;
                          WAPP.CPP  7-22-95  11:36a
```

```
                HDC hdc = GetDC( hvu );

                int noiseright, speechleft, speechright, unit;

                unit = (rc.right - rc.left) / fepinfo.max_level;

                noiseright = speechleft = unit * fepinfo.noise_level;

                speechright = unit * fepinfo.speech_level + speechleft;

                //      draw speech
                HBRUSH hBrush = CreateSolidBrush( RGB( 255, 0, 0 ) );

                HBRUSH hOldBrush = (HBRUSH )SelectObject( hdc, hBrush );

                Rectangle( hdc, rc.left, rc.top, noiseright, rc.bottom );

                SelectObject( hdc, hOldBrush );

                DeleteObject(hBrush);

                //      draw speech
                hBrush = CreateSolidBrush( RGB( 0, 255, 0 ) );

                hOldBrush = (HBRUSH )SelectObject( hdc, hBrush );

                Rectangle( hdc, speechleft, rc.top, speechright, rc.bottom );

                SelectObject( hdc, hOldBrush );

                DeleteObject( hBrush );

                // Background
                Rectangle( hdc, speechright, rc.top , rc.right, rc.bottom );

                ReleaseDC( hvu, hdc );
        )
)

void DlgWin::onDestroy( HWND hwnd )
(
        KillTimer( hwnd, VUTIMER_ID );

        if( speechTask )
        (
                zipUtt.killUtt();
                cityUtt.killUtt();

                if( channel )
                (
                        delete channel;
                        channel = 0;
                )

                if( recog )
                (
                        delete recog;
                        recog = 0;

                        choiceListStats.printStats();
                )

                delete speechTask;
                speechTask = 0;
                                Page 26 of 29
        }

        for( int i=0; i<4; ++i )
        {
                if( comm[ i ] )
                {
                        delete comm[ i ];

                        comm[ i ] = 0;
                }
                if( commformat[ i ] )
                {
                        delete commformat[ i ];

                        commformat[ i ] = 0;
                }
        }

        //    PostQuitMessage( 0 );
}



BOOL FAR PASCAL _export DlgWin::wndProc( HWND hwnd, UINT message, WPARAM wParam,
  =>  LPARAM lParam )
{
        messageHandled = TRUE;

        long returnValue = 0;

        switch( message )
        {
                handleMessage( WM_INITDIALOG, hwnd, wParam, lParam, onInitDialog
  =>  );
                handleMessage( WM_DESTROY, hwnd, wParam, lParam, onDestroy );
                handleMessage( WM_COMMAND, hwnd, wParam, lParam, onCommand );
                handleMessage( WM_SYSCOMMAND, hwnd, wParam, lParam, onSysCommand
  =>  );
                handleMessage( WM_TIMER, hwnd, wParam, lParam, onTimerDrawVU );
                handleMessage( UM_CHANNELSTART, hwnd, wParam, lParam, onSpeech )
  => ;
                handleMessage( UM_COMM_CHECK_MSG, hwnd, wParam, lParam, onCommCh
  => eckMsg );
                handleMessage( UM_COMM_SEND_SYNC_MSG, hwnd, wParam, lParam, onCo
  => mmSendSyncMsg );
        }

        return messageHandled ? returnValue : 0;
}



/////////////////////////////////////////////////////////////////////////////
//
// CLASS AppWin
//
// Purpose : application driven from this class
//                      WinMain will call wndProc of Appwin
//
//
/////////////////////////////////////////////////////////////////////////////

class AppWin
{
```

```
public:
        static long FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );

        static HWND      hWnd;
        static HTASK hTask;

protected:

        static bool messageHandled;

        // wndProc() services
        static bool onCreate( HWND hWnd, CREATESTRUCT FAR* lpCreateStruct );
        static void onDestroy( HWND hWnd );
        static void onNCDestroy( HWND hWnd );
        static void onClose( HWND hWnd );
        static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y );


        friend int PASCAL WinMain( HINSTANCE, HINSTANCE, LPSTR, int );
};


// Static Global Variables of AppWin
HWND            AppWin::hWnd;
HTASK           AppWin::hTask;
bool            AppWin::messageHandled;

/////////////////////////////////////////////////////////////////////////////

Main function of application

/////////////////////////////////////////////////////////////////////////////


int PASCAL WinMain( HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR lpszCmdLine,
  =>  int /* nCmdShow */ )
{
        static const char name[] = "DragCpp";

        SetMessageQueue( 64 );

        if( !hPrevInstance )
        {
                WNDCLASS wndclass;

                wndclass.style          = CS_HREDRAW | CS_VREDRAW;
                wndclass.lpfnWndProc    = AppWin::wndProc;
                wndclass.cbClsExtra     = 0;
                wndclass.cbWndExtra     = 0;
                wndclass.hInstance      = hInst;
                wndclass.hIcon          = LoadIcon( 0, MAKEINTRESOURCE( IDI_MY_IC
  => ON ) );
                wndclass.hCursor        = LoadCursor( 0, IDC_ARROW );
                // wndclass.hIcon          = LoadIcon( hInst, "merc.ico" );
                // wndclass.hCursor        = LoadCursor( 0, IDC_ARROW );
                wndclass.hbrBackground = 0;
                wndclass.lpszMenuName  = 0;
                wndclass.lpszClassName = name;

                RegisterClass( &wndclass );
        }

        ::hInstance = hInst;    // the global hInstance for the sliderctl
        slideControl.libMainInit();     // register the slider class

        HWND hWnd = CreateWindow( name,
=>                              name,
=>                              WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
=>                              CW_USEDEFAULT,
=>                              CW_USEDEFAULT,
=>                              CW_USEDEFAULT,
=>                              CW_USEDEFAULT,
=>                              0,
=>                              0,
=>                              hInst,
=>                              lpszCmdLine );

        assert( hWnd == AppWin::hWnd );

        ShowWindow( AppWin::hWnd, SW_HIDE );
        //      ShowWindow( hWnd, nCmdShow );
        //      UpdateWindow( hWnd );

        MSG msg;

        while( GetMessage( &msg, 0, 0, 0 ) )
        {
                TranslateMessage( &msg );
                DispatchMessage( &msg );
        }

        return msg.wParam;
}

// Application Helper function

void AppWin::onCommand( HWND hWnd, UINT cmd, HWND, UINT )
{
        switch ( cmd )
        {
         case WM_INIT_DIALOG:
                {
                        // create dialog box proc instances for use later
                        static FARPROC lpfn = MakeProcInstance( (FARPROC )DlgWin
  => ::wndProc, hInstance );

                        assert( lpfn );

                        DialogBox( hInstance, "TestDialog", 0, (DLGPROC )lpfn );

                        PostMessage( hWnd, WM_CLOSE, 0, 0 );
                }
                break ;


         default:
                messageHandled = FALSE;
        }
}
```

```
// Application Helper function

bool AppWin::onCreate( HWND hwnd, CREATESTRUCT FAR* )
{
        hTask = GetCurrentTask();

        hWnd = hwnd;

        PostMessage( hwnd, WM_COMMAND, WM_INIT_DIALOG, 0 );

        return TRUE;
}

// Application Helper function

void AppWin::onDestroy( HWND )
{
        PostQuitMessage( 0 );
}


// Application Helper function

void AppWin::onNCDestroy( HWND )
{
}


// Application Helper function

void AppWin::onClose( HWND hWnd )
{
        DestroyWindow( hWnd );
}


// Application message handler called from winMain function

long FAR PASCAL _export AppWin::wndProc( HWND hwnd, UINT message, WPARAM wParam,
  =>  LPARAM lParam )
{
        messageHandled = TRUE;

        long returnValue = 0;

        switch( message )
        {
                handleMessage( WM_CREATE, hwnd, wParam, lParam, onCreate );
                handleMessage( WM_DESTROY, hwnd, wParam, lParam, onDestroy );
                handleMessage( WM_NCDESTROY, hwnd, wParam, lParam, onNCDestroy )
  => ;
                handleMessage( WM_CLOSE, hwnd, wParam, lParam, onClose );
                handleMessage( WM_COMMAND, hwnd, wParam, lParam, onCommand );

         default:
                messageHandled = FALSE;
                break;
        }
        return( messageHandled
                                        ? returnValue
                                        : DefWindowProc( hwnd, message, wParam,
  => lParam ) );
}
```

WAPP.CPP 7-22-95 11:36a

Page 29 of 29

```
/*****************************************************************************
        Description

        oneshot.cpp
        application's memory manager. The app knows and controls where to
        get memory for allocation. It comes with blocks. Advantage
        to get rid of allocated memory and to make the app more robust against
        memory leaks.

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 - 1995
*****************************************************************************/


#include "oneshot.h"


OneShotHeap::OneShotHeap( unsigned sz )
: Seq< MemSL >()
{
    start=end=0;        heapChunkSize = sz;
}


OneShotHeap::~OneShotHeap()
{
    for( MemSL *mslNext, *msl=first(); msl; msl=mslNext)
    {
        mslNext = msl->next();
        delete msl;
    }
    f=l=0;
}


void* OneShotHeap::carveItUpAndGiveItAway( unsigned length )
{
    // our own little new...

    length += length & 1;       // align word boundry (even addresses)

    char* st = start;

    if( end - st < length )
    {
        unsigned sz = length + sizeof( MemSL ) + ( sizeof( MemSL ) & 1 );

        if( sz < heapChunkSize )        sz = heapChunkSize;

        st = new char[ sz ];

        assert( st );

        end = st + sz;

        addFirst( (MemSL* )st );

        st += sizeof( MemSL ) + ( sizeof( MemSL ) & 1 ); // align word boundry
    }
    start = st + length;

    return (void* )st;
```

ONESHOT.CPP 7-22-95 11:24a

```
}

void OneShotHeap::empty()
{
    MemSL* msl = first();

    if( msl == 0 )
        return;

    MemSL* mslNext;

    while( (mslNext=msl->next()) != 0 )
    {
        remove( msl );

        delete (char* ) msl;

        msl = mslNext;
    }

    start = (char* )msl;

    end = start + heapChunkSize;

    start += sizeof( MemSL );
}
```

Page 1 of 1

```
/*************************************************************************
        Description

        Hypo.cpp
        Module that computes zip hypotheses. Stack decoder algoritm
        is implemented in this module.

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995

*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <assert.h>
#include <ctype.h>

#include "hashcsz.h"


/////////////////////////////////////////////////////////////////////////
// ZipResult member functions
/////////////////////////////////////////////////////////////////////////
char stateAliasString[ 40 ] = "michigan";

ZipResult::ZipResult( ZipResult& zr )
{
    memcpy( (void* )this, (void* )&zr, sizeof( ZipResult ) );
}


// Given the phrase, compute where
// start index for zip is, and index for state is within the phrase

int ZipResult::initFromPhrase()
{
    char *ph = phrase;
    char *spl = spelling;

    stateSpellingIndex = -1;
    zipStateSpellingIndex = -1;

    while( *ph && !isdigit( *ph ) )
    {
        *spl++ = *ph++;
    }

    for( int len = 0; *ph != '\0'; )
    {
        if( isdigit( *ph ) )
        {
            if( zipStateSpellingIndex == -1 )
                zipStateSpellingIndex = spl - spelling;

            *spl++ = *ph++;

            ++len;
        }

        else if( isspace( *ph ) )
            ++ph;
                        HYPO.CPP  7-22-95  10:24a
```

```
        else
        {
            if( spl != spelling )               *spl++ = ' ';

            stateSpellingIndex = spl - spelling;

            while( (*spl++ = *ph++) != '\0' )
                ;
            break;
        }
    }

    *spl = '\0';

    if( zipStateSpellingIndex == -1 )
        zipStateSpellingIndex = spl - spelling;

    if( stateSpellingIndex == -1 )
        stateSpellingIndex = spl - spelling;

    else
    {
        ++len;

        if( 0 == strcmp( stateName(), "state" ) )
            strcpy( stateName(), stateAliasString );
    }

    return( len );
}

// Given the spelling, compute where
// start index for zip is, and index for state is within the spelling

int ZipResult::initFromSpelling()
{
    char *ph = phrase;
    char *spl = spelling;

    stateSpellingIndex = -1;
    zipStateSpellingIndex = -1;

    while( *spl && !isdigit( *spl ) )
    {
        *ph++ = *spl++;
    }

    for( int len = 0; *spl != '\0'; )
    {
        if( isdigit( *spl ) )
        {
            if( zipStateSpellingIndex == -1 )
                zipStateSpellingIndex = spl - spelling;

            *ph++ = *spl++;

            *ph++ = ' ';

            ++len;
        }
        else
            break;
    }

    while( isspace( *spl ) )
                        Page 1 of 5
        ++spl;

    stateSpellingIndex = spl - spelling;

    if( *spl )
    {
        ++len;

        while( *spl )
            *ph++ = *spl++;
    }

    *ph = '\0';

    if( zipStateSpellingIndex == -1 )
        zipStateSpellingIndex = spl - spelling;

    else
    {
        if( 0 == strcmp( stateName(), "state" ) )
            strcpy( stateName(), stateAliasString );
    }

    return( len );
}


/////////////////////////////////////////////////////////////////////
// ZipHypothesis and ZipHypoAC member functions
/////////////////////////////////////////////////////////////////////

// Delete new words, usually phrase built
void ZipHypoAC::clearUpNewWords( Recognizer* recog )
{
    for( int i=count(); i--; )
    {
        if( (*this)[ i ].isNewWord )
        {
            recog->deleteWord( (*this)[ i ].wordId );

            (*this)[ i ].isNewWord = 0;
        }
    }
}

// convert zip to a string
const char* ltoZip( long num )
{
    static char buf[ UTT_PROMPT_LENGTH ] = "00000";

    ltoa( num, buf+5, 10 );

    return buf + strlen( buf ) - 5;
}


ZipHypothesis::ZipHypothesis( ZipHypothesis& zr )
{
    memcpy( (void* )this, (void* )&zr, sizeof( ZipHypothesis ) );
}


// initialize ZipHypothesis
```

```
// 3 parameters: recognizer , the zip and the ZipResult class

ZipHypothesis::ZipHypothesis( Recognizer* recog, long num, ZipResult& zr )
: ZipResult( zr )
{
        // convert digit code  to a string
    const char *zp = ltoZip( num );

        // copy the zip string into the state name
    strncpy( zipStateName(), zp, 5 );


    char *s = phrase + zipStateSpellingIndex;

    while( *zp )
    {
        *s++ = *zp++;

        ++s;
    }

        // set boolean to 1 if the phrase does not have a word ID yet.
        // If this word does not exists yet, most often this is the case,
        // create a new model and add it to the state. Look in dragcpp.cpp
        // for buildWord() and addWord()

    isNewWord = ( (wordId=recog->wordId( spelling )) == 0 );

        // buildWord builds and adds the WORD_ID to the state
    if( isNewWord )
        wordId = recog->buildWord( spelling, phrase );

    else
        recog->addWord( wordId );

    assert( wordId != 0 );
}

// initialize ZipHypothesis
// 2 parameters: recognizer , the zip and the ZipResult class

ZipHypothesis::ZipHypothesis( Recognizer* recog, ZipResult& zr )
: ZipResult( zr )
{
    isNewWord = ( (wordId=recog->wordId( spelling )) == 0 );

    if( isNewWord )
        wordId = recog->buildWord( spelling, phrase );

    else
        recog->addWord( wordId );

    assert( wordId != 0 );
}


/////////////////////////////////////////////////////////////////////
// Hypothesis member functions/data
/////////////////////////////////////////////////////////////////////

int compareHypothesis( const void* given, const void* test )
{
    return ((Hypothesis* )test)->currentScore - ((Hypothesis* )given)->currentSc
  => ore;
}
```

```
// confusability matrix used in the stack decoder
int Hypothesis::confusionScore[] = {
        // for DEBUGGING
//      0, 800, 800, 800, 800, 800, 800, 800, 800, 800,
//    800,   1, 800, 800, 100, 800, 800, 800, 800, 800,
//    800, 800,   2, 800, 800, 800, 800, 800, 800, 800,
//    800, 800, 800,   3, 800, 800, 800, 800, 800, 800,
//    800, 800, 800, 800,   4, 800, 800, 800, 800, 800,
//    800, 800, 800, 800, 800,   5, 800, 800, 800, 150,
//    800, 800, 800, 800, 800, 800,   6, 800, 800, 800,
//    800, 800, 800, 800, 800, 800, 800,   7, 800, 800,
//    800, 800, 800, 800, 800, 800, 800, 800,   8, 800,
//    800, 800, 800, 800, 800, 125, 800, 800, 800,   9,

        0,  70,  54,  60,  71,  69,  70,  68, 160, 155,
       75,   0, 161, 161,  43,  47, 162,  57, 160,  63,
       47,  70,   0,  30,  58,  64,  38,  53,  57,  63,
       84, 162,  54,   0, 163,  64,  64, 161,  51,  54,
       75,  43,  69,  69,   0,  49, 162,  51, 160,  63,
       84,  70, 161,  63,  71,   0, 162, 161, 160,  36,
       94, 162,  64,  60, 163, 161,   0, 161, 160, 155,
       84, 162,  69,  69,  71,  64, 162,   0, 160,  63,
       75,  65,  61,  47, 163, 161,  64, 161,   0, 155,
      166, 162,  69, 161, 163,  64, 162, 161,  68,   0,
};


// memory manager
OneShotHeap Hypothesis::heap( 4096 );

// function that generates  new hypotheses and puts them in the
// priority queue
void Hypothesis::propagate( PriorityQue<Hypothesis>* hpq, int stateId )
{
    int newScore[ 10 ];

    int flag = 0;

    int currentIndex = nextWordIndex++;

    int oldScore = currentScore;

        // number of digits
    for( int i=0; i<10; ++i )
    {
        // first do Language Modeling...
        // If zips do not make sense for recognized state name, ignore
        // hypothesis
        if( currentIndex == 0 )
        {
            if( !StateZip::isFirstDigitLegal( i, stateId ) )
                continue;
        }
        else if( currentIndex == 1 )
        {
            if( !StateZip::areFirstTwoDigitsLegal( wordHistoryArray[ 0 ], i, sta
  => teId ) )
                continue;
        }
        else if( currentIndex == 2 )
        {
            if( !StateZip::areFirstThreeDigitsLegal( wordHistoryArray[ 0 ],
                                            wordHistoryArray[ 1 ], i, stateId ) )
                continue;
        }
```

HYPO.CPP 7-22-95 10:24a

```
//      else if( currentIndex == 4 )
//      {
//          if( !StateZip::areFirstFiveDigitsLegal( wordHistoryArray[ 0 ],
//                                          wordHistoryArray[ 1 ], i, stateId ) )
//              continue;
//      }

        // Language model OK's it...


        newScore[ i ] = ( oldScore +
                      confusionScore[ 10*data[ currentIndex ] + i ] );

        // only consider candidates below a score of 300
        if( newScore[ i ] < 300 )
        {
            Hypothesis *hypo = ( flag == 0 ?
                                 this :
                                 new( &heap ) Hypothesis( *this ) );

            flag = 1;

            hypo->wordHistoryArray[ currentIndex ] = i;

            hypo->currentScore = newScore[ i ];

                // put hypothesis into priority queue
            hpq->push( hypo );
        }
    }
}


// Function called from within onSpeech() in wapp.cpp
// Stack decoder algoritm

// numWords corresponds to wasGoodZipPhrase in the onSpeech function.
// ZipHypoAC is a growable array. It inherits from ZipHypothesis


int buildHypothesis( Recognizer* recog, ZipHypoAC* choiceList,
                     ZipResult& zr, int numWords  )
{
    static PriorityQue< Hypothesis > topList( compareHypothesis );
    static PriorityQue< Hypothesis > hypoPQ( compareHypothesis );

    topList.removeAll();
    hypoPQ.removeAll();
    Hypothesis::heap.empty();


        // data contains digits and state name
    int data[ 20 ];

        // Get zip and state name from ZipResults zr
        char* zp = zr.zipStateName();

    data[ 0 ] = zp[ 0 ] - '0';
    data[ 1 ] = zp[ 1 ] - '0';
    data[ 2 ] = zp[ 2 ] - '0';
    data[ 3 ] = zp[ 3 ] - '0';
    data[ 4 ] = zp[ 4 ] - '0';


        // the class Hypothesis is used to generate the hypotheses. Once
```

Page 3 of 5

```
    // we have those, we will have to convert the information to
    // a zipHypothesis class, which contains info about the recognizer's
    // opinion on the hypothezised digit phrases

Hypothesis *hypo = 0;


    /* If number of words returned from recognizer is 5 instead of 6
       do some corrections:

    1. If the fifth word (4 index) is an 8 or a 3, it is likely to be
    the state name that is misrecognized. Replace that word "8" or "3"
    with the place holder string " state".


    2. Only apply 1. if the word "8" appears in any other position
    in the zip phrase.

    This is the only case where we noticed a regular deletion error.
    If there is no "8" in the preceding     4 digits, we cannot reliably
    change 4 digits into 5 with hope of success.

    */


if( numWords == 5 &&
    (zp[ 4 ] == '8' || zp( 4 ) == '3') &&
    (zp[ 0 ] == '8' || zp( 1 ) == '8' || zp( 2 ) == '8' || zp[ 3 ] == '8') )
(
    // fixing 88 -> 8 and "state" -> 8 or 3
    memcpy( &(data[5]),  data, 5*sizeof( int ) );
    memcpy( &(data[10]), data, 5*sizeof( int ) );
    memcpy( &(data[15]), data, 5*sizeof( int ) );

    int *s=data, *t=data+5, *tstart=data+5, *send=data+4;

    int dup = 0;

    while( s<send )
    (
        ++t;

        if( *s++ == 8 )
        (
            while( *s == 8 && s<send )
            (
                ++s;
                ++t;
            )
            *t = 8;

            memcpy( t+1, s, (send - s) * sizeof( int ) );

            // initialize priorityQueue with first hypothesis: == the correc
=> ted
            // digit string

            //  stack decoder will be used a little later

            hypo = new( &Hypothesis::heap ) Hypothesis;
            hypo->currentScore = 0;
            hypo->data = tstart;
            hypoPQ.push( hypo );

            if( dup == 0 )
            (
                        HYPO.CPP  7-22-95  10:24a

                dup = 1;

                zp[ 0 ] = tstart[ 0 ] + '0';
                zp[ 1 ] = tstart[ 1 ] + '0';
                zp[ 2 ] = tstart[ 2 ] + '0';
                zp[ 3 ] = tstart[ 3 ] + '0';
                zp[ 4 ] = tstart[ 4 ] + '0';
                zp[ 4 ] = tstart[ 4 ] + '0';

                    // fill in place holder " state"
                strcpy( zp + 5, " state" );

                numWords = zr.initFromSpelling();
            )

            t += 5;
            tstart += 5;
        )
    )
)


    // Normal case: 6 words were returned by contRecog : 5 digits and state
=> name

else if( numWords == 6 )
(
    hypo = new( &Hypothesis::heap ) Hypothesis;
    hypo->currentScore = 0;
    hypo->data = data;
    hypoPQ.push( hypo );
)

if( hypo )
(

    // get all zips for state name

    StateToStateZipSL* ssz = StateZip::findStateToStateZipSL( zr.stateName()
=>  );

    if( ssz )
    (
            // index into table of type StateZip : which file??
        int stateId = ssz->useIndex;


            // stack decoder at work until 40 most likely digit-string confu
=> sions
            // have been found
        while( topList.count() < 40 &&
            0 != (hypo=hypoPQ.pop()) )
        (
            if( hypo->nextWordIndex == 5 )
                topList.push( hypo );

            else
                hypo->propagate( &hypoPQ, stateId );
        )
    )
)
// now fill ChoiceList

int cnt=topList.count(); // should be 40
                                Page 4 of 5
        // Get memory for ChoiceList (of type ZipHypAC:ZipHypothesis)

    if( cnt )
    {
        int i = choiceList->makeRoom( cnt );

        while( cnt-- )
        {
                // process the data from topList[] class into a ZipHypothesis ge
»  tting
                // SD_WORD id's for the digit strings, and move it into a ZipHyp
»  oAC
                // the class for the choice list.
                // After this, we have our ContRecog choice list.

                // Have a look at ZipHypothesis::ZipHypothesis

            new( (void* )( &( (*choiceList)[ i++ ] ) ) )
            ZipHypothesis( recog, topList[ cnt ]->value(), zr );
        }
    }


        // There is only one hypothesis, because the correction from 5 to 6
        // failed. Happens very rarely.

    if( !choiceList->count() || strcmp( (*choiceList)[ 0 ].spelling, zr.spelling
»    ) )
    {
        int j = choiceList->makeRoom( 1 );

        new( (void* )( &( (*choiceList)[ j ] ) ) ) ZipHypothesis( recog, zr );
    }

        // return 6, or 5 if correction of deletion error failed.
    return numWords;
}
```

```
////////////////////////////////////////////////////////////////////////////
/*

    Description

        WENROLL.CPP

        Training app to train USR files to user's voice. Not done in noise.

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995

*/

#include <windowsx.h>
#include <string.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <io.h>

#include <fcntl.h>
#include <ctype.h>

#include "dragcpp.h"
#include "wenrol.h"
#include "slidectl.h"


// this is the instance of the control
DgnSlideControl slideControl;
extern HINSTANCE hinstance;      // when we call DgnSlideControl::libMainInit()
HINSTANCE hinstance;             // hinstance must be initialized...


static char femCityVoc[]         = "ctall.vof";
static char femDigitsVoc[]       = "digits.vcf";
static char malCityVoc[]         = "ctall.vom";
static char malDigitsVoc[]       = "digits.vom";
static char* cityVoc = 0;
static char* digitsVoc = 0;


#define LINE_SIZE 80
char usrName[ LINE_SIZE ];

#define handleMessage( message, hWnd, wParam, lParam, fn ) \
    case (message) : returnValue = HANDLE_ ## message( (hWnd), (wParam), ((lParam
  => ), (fn) ); break;



 class EnrollDlgWin
 {
  public:
     static long FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );
     static HWND  hWnd;

  protected:

     static bool messageHandled;
     static bool sawError;
     static bool ignoreErrors;
     static char* name;
```

WENROL.CPP 7-22-95 11:36a

```
    static short wordPause;
    static bool useComboSelect;
    static bool isMale;

    // wndProc() services
    static bool onInitDialog( HWND hWnd, HWND topCntrl, long );
    static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y );
};

HWND             EnrollDlgWin::hWnd;
bool             EnrollDlgWin::messageHandled;
bool             EnrollDlgWin::sawError;
bool             EnrollDlgWin::ignoreErrors;
char*            EnrollDlgWin::name = "dragon123\0      ";
short            EnrollDlgWin::wordPause=256;
bool             EnrollDlgWin::useComboSelect;
bool             EnrollDlgWin::isMale = 1;

bool EnrollDlgWin::onInitDialog( HWND hDlg, HWND, long )
{
    hWnd = hDlg;

    struct ffblk attrib;

    // set up the usr listbox
    if( !findfirst( "*.uq?", &attrib, FA_NORMAL ) )
    {
        do {
            SendDlgItemMessage( hDlg, ENROL_USR_COMBO_ID, CB_ADDSTRING, 0, (DWOR
  => D) attrib.ff_name );
        } while( !findnext( &attrib ) );
    }

    CheckRadioButton( hDlg, RADIO_MALE_ID, RADIO_FEMALE_ID, RADIO_MALE_ID );

    return TRUE;
}


void EnrollDlgWin::onCommand( HWND hwnd, UINT cmd, HWND, UINT childCmd )
{
    switch ( cmd )
    {
     case ENROL_USR_COMBO_ID:
        if( childCmd == CBN_SELENDOK )
        {
            if( !GetDlgItemText( hwnd, ENROL_USR_COMBO_ID, usrName, LINE_SIZE )
  => )
                return;

            SetWindowText( GetDlgItem( hwnd, EDIT_SSNUM_ID ), "" );
            SetWindowText( GetDlgItem( hwnd, EDIT_SHIFT_ID ), "" );

            useComboSelect = 1;
        }
        break;

     case EDIT_SSNUM_ID:
        if( childCmd == EN_SETFOCUS )
            useComboSelect = 0;
        break;

     case EDIT_SHIFT_ID:
        if( childCmd == EN_SETFOCUS )
            useComboSelect = 0;
        break;
```

Page 1 of 8

```
        case RADIO_MALE_ID:
        case RADIO_FEMALE_ID:
            if( childCmd == BN_CLICKED )
            {
                CheckRadioButton( hwnd, RADIO_MALE_ID, RADIO_FEMALE_ID, cmd );
                useComboSelect = 0;
            }
            break;

            //      case CREATING_USER_TEXT_ID:
            //       break;

        case ENROL_BTN_ID:
            if( useComboSelect )
            {
                if( GetDlgItemText( hwnd, ENROL_USR_COMBO_ID, usrName, LINE_SIZE ) )
                    EndDialog( hwnd, 1 );
                else
                    beep( "bad.wav" );

                return;
            }
            else
            {
                usrName[ 0 ] = '\0';

                GetDlgItemText( hwnd, EDIT_SHIFT_ID, usrName, LINE_SIZE );

                if( !isdigit( usrName[ 0 ] ) )
                {
                    beep( "bad.wav" );
                    return;
                }

                usrName[ 1 ] = ( SendDlgItemMessage( hwnd, RADIO_MALE_ID, BM_GETCHEC
=> K, 0, 0 )
                                  ? 'M'
                                  : 'F' );

                usrName[ 2 ] = '\0';

                GetDlgItemText( hwnd, EDIT_SSNUM_ID, usrName +2, LINE_SIZE -2 );

                char *s = usrName + strlen( usrName );

                if( s - usrName < 6 )
                {
                    beep( "bad.wav" );
                    return;
                }

                s -= 4;

                for( char* ss = usrName + 2; (*ss++ = *s++) != 0; );

                for( s=usrName; isalnum( *s ); ++s );

                if( *s != '\0' )
                {
                    beep( "bad.wav" );
                    return;
                }
                *s++ = '.';
                *s++ = 'u';
                *s++ = 'q';
```

MENROL.CPP 7-22-95 11:36a

```
                *s++ = usrName[ 1 ];
                *s++ = '\0';

                int wfd = open( usrName, O_WRONLY | O_BINARY );

                if( wfd == -1 )
                {
                    SetWindowText( GetDlgItem( hwnd, CREATING_USER_TEXT_ID ), "Creat
=> ing New User" );

                    int rfd = open( ( usrName[ 1 ] == 'M'
                                      ? "base.usm"
                                      : "base.usf" ), O_RDONLY | O_BINARY );

                    if( rfd == -1 )
                    {
                        MessageBox( hwnd, "Could not open one of base.usm or base.us
=> f",
                                    "Enrollment Error", MB_TASKMODAL | MB_ICONSTOP |
=>  MB_OK );

                        EndDialog( hwnd, 0 );
                    }

                    wfd = creatnew( usrName, FA_NORMAL );
                    close( wfd );
                    wfd = open( usrName, O_WRONLY | O_BINARY );

                    if( wfd == -1 )
                    {
                        MessageBox( hwnd, "Could not create a new usr",
                                    "Enrollment Error", MB_TASKMODAL | MB_ICONSTOP |
=>  MB_OK );

                        EndDialog( hwnd, 0 );
                    }

                    int bytesRead;
                    char buf[ 2 * 1024 ];

                    HCURSOR hCursor = SetCursor( LoadCursor( 0, IDC_WAIT ) );

                    while( (bytesRead = read( rfd, buf, sizeof(buf) ) ) != 0 )
                    {
                        while( bytesRead )
                            bytesRead -= write( wfd, buf, bytesRead );
                    }

                    SetCursor( hCursor );

                    close( wfd );

                    close( rfd );

                    EndDialog( hwnd, 1 );
                }
            }
            break;

        case ENROL_EXIT_BTN_ID:
            EndDialog( hwnd, 0 );
            break;
    };
}

long FAR PASCAL _export EnrollDlgWin::wndProc( HWND hWnd, UINT message, WPARAM w
```

Page 2 of 8

```
  => Param, LPARAM lParam )
{
    messageHandled = TRUE;

    long returnValue = 0;

    switch( message )
    {
        handleMessage( WM_INITDIALOG, hWnd, wParam, lParam, onInitDialog );
        handleMessage( WM_COMMAND, hWnd, wParam, lParam, onCommand );
    }

    return returnValue;
}

//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////


class TrainDlgWin
{
 public:
    static long FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );
    static HWND  hWnd;

 protected:

    static bool messageHandled;
    static bool sawError;
    static bool ignoreErrors;
    static char* name;
    static short wordPause;

    static SpeechTask *speechTask;                      // speech driver task ID
    static UttChannel *channel;
    static Recognizer* recog;

    static void reportError( int code, char far *message );
    static void SD_CALLBACK _export postSpeechEvent( SD_CHANNEL, SD_CHANNEL_EVEN
  => T );
    static void setupPrompt( HWND hWnd );

    // wndProc() services
    static bool onInitDialog( HWND hWnd, HWND topCntrl, long );
    static void onDestroy( HWND hWnd );
    static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y );
    static void onSpeech( HWND hWnd, SD_CHANNEL ch );
};

HWND            TrainDlgWin::hWnd;
bool            TrainDlgWin::messageHandled;
bool            TrainDlgWin::sawError;
bool            TrainDlgWin::ignoreErrors;
char*           TrainDlgWin::name = "dragon123\0      ";
short           TrainDlgWin::wordPause=256;
SpeechTask*     TrainDlgWin::speechTask;
Recognizer*     TrainDlgWin::recog;
UttChannel*     TrainDlgWin::channel;


bool TrainDlgWin::onInitDialog( HWND hDlg, HWND, long )
```

WENROL.CPP 7-22-95 11:36a

```
{
    hWnd = hDlg;

    speechTask = new SpeechTask( name, reportError );

    recog = 0;

    struct ffblk attrib;

    // set up the usr listbox
    if( !findfirst( "*.uq?", &attrib, FA_NORMAL ) )
    {
        do {
            SendDlgItemMessage( hDlg, USR_COMBO_ID, CB_ADDSTRING, 0, (DWORD) att
  => rib.ff_name );
        } while( !findnext( &attrib ) );
    }

    // set up the prompt listbox
    if( !findfirst( "*.txt", &attrib, FA_NORMAL ) )
    {
        do {
            SendDlgItemMessage( hDlg, PROMPT_COMBO_ID, CB_ADDSTRING, 0, (DWORD)
  => attrib.ff_name );
        } while( !findnext( &attrib ) );
    }

    speechTask->setPar( "adapt-tolerance", (short )3 );

    speechTask->setPar( "quick-talk", (short )0 );

    speechTask->setPar( "computation", (short )1 );

    SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLRANGE, 0, 0x02000
  => 064L );

    SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLPOS, 0x0100, 1 );

    return TRUE;
}


void TrainDlgWin::reportError( int code, char far *message )
{
    sawError = YES;
    if( ignoreErrors )
        return;          // prevent recursion
    ignoreErrors = YES;

    // Throw up an error message
    char buffer[ 512 ];
    sprintf( buffer, "An error has occurred\n  code = %d\n  message = %s",
             code, message );

    if( stricmp( message, "Bad token" ) )
    {
        MessageBox( hWnd, buffer, "WinTalk SDAPI Error",
                    MB_TASKMODAL | MB_ICONSTOP | MB_OK );

        // terminate...  kill the hTopParentWindow
        if( hWnd )
            PostMessage( hWnd, WM_CLOSE, 0, 0 );
    }
    ignoreErrors = 0;
}
```

Page 3 of 8

```
void SD_CALLBACK _export TrainDlgWin::postSpeechEvent( SD_CHANNEL ch, SD_CHANNEL
  => _EVENT eventIn )
{
    assert( eventIn == SD_CHANNEL_START );

    MSG msg;

    if( 0 == PeekMessage( &msg, hWnd, UM_CHANNELSTART, UM_CHANNELSTART, PM_NOREM
  => OVE | PM_NOYIELD ) )
        PostMessage( hWnd, UM_CHANNELSTART, (WORD )ch, 0 );

} // EventHandler

#define HANDLE_UM_CHANNELSTART(hwnd, wParam, lParam, fn) \
                              ((fn)(hwnd, (SD_CHANNEL )wParam), 0L)


int numDigits( const char* s )
{
    for( int len = 0; *s; ++s )
    {
        if( isdigit( *s ) )
            ++len;

        else if( !isspace( *s ) )
            return 0;
    }
    return len;
}

void TrainDlgWin::setupPrompt( HWND hWnd )
{
    char zipBufP[ LINE_SIZE ];
    char stateBufP[ LINE_SIZE ];
    char cityBufP[ LINE_SIZE ];

    int i = recog->promptIndex() % 3;        // -1 == EOF

    int promptOffsets[] = { -2, -1, 0, 1, 2 };

    int* peekOffset = promptOffsets + 2 - i;

    if( i != -1 )
    {
        recog->peekPrompt( zipBufP, LINE_SIZE, peekOffset[ 0 ] );

        recog->peekPrompt( stateBufP, LINE_SIZE, peekOffset[ 1 ] );

        recog->peekPrompt( cityBufP, LINE_SIZE, peekOffset[ 2 ] );
    }
    else
    {
        zipBufP[ 0 ] = '\0';
        stateBufP[ 0 ] = '\0';
        cityBufP[ 0 ] = '\0';

        if( i == -1 && recog->isListening() )
            PostMessage( hWnd, WM_COMMAND, MICROPHONE_BTN_ID, 0 );

        else i = -1;
    }

    if( !recog->isListening() )
        i = -1;
```

WENROL.CPP 7-22-95 11:36a

```
    SetWindowText( GetDlgItem( hWnd, PROMPT_ZIP_TXT_ID ), zipBufP );

    SetWindowText( GetDlgItem( hWnd, PROMPT_STATE_TXT_ID ), stateBufP );

    SetWindowText( GetDlgItem( hWnd, PROMPT_CITY_TXT_ID ), cityBufP );


    SetWindowText( GetDlgItem( hWnd, SAY_ZIP_TXT_ID ),
                   ( i == 0 ? "Say:" : "" ) );
    SetWindowText( GetDlgItem( hWnd, SAY_STATE_TXT_ID ),
                   ( i == 1 ? "Say:" : "" ) );
    SetWindowText( GetDlgItem( hWnd, SAY_CITY_TXT_ID ),
                   ( i == 2 ? "Say:" : "" ) );
}

void TrainDlgWin::onSpeech( HWND hwnd, SD_CHANNEL ch )
{
    // get the utterance handle; do not wait for the end of the
    // utterance since the recognition can proceed in parallel
    // with the utterance collection

    assert( recog );

    if( recog->notifyChannel( ch, SD_CHANNEL_START ) )
    {
        char buf[ LINE_SIZE ],

        SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_RESETCONTENT, 0, 0 );

        const char* s = recog->prompt();

        if( s == 0 )
        {
            recog->killUtt();
            return;
        }

        int digitLen = numDigits( s );

        static AC< SD_WORD > passThru;

        if( digitLen )
        {
            recog->setVoc( digitsVoc );
            recog->setState( "digits" );
            recog->seperateDigits( digitLen );
        }
        else
        {
            recog->setVoc( cityVoc );
            recog->setState( (SD_STATE )0 );

            passThru.removeAll();
            SD_WORD promptID = recog->wordId( s );
            if( promptID )
                passThru.add( promptID );
        }

        if( 0 == ( digitLen ? recog->contRecog( digitLen ) : recog->recog( &pass
  => Thru ) ) )
        {
            char buf[ LINE_SIZE ], *b;

            int correct = -1;
```

Page 4 of 8

```
        for( int i = 0; i < recog->resultCount(); ++i )
        {
            recog->resultName( buf, sizeof( buf ), i );

            SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADDSTRING, 0, (DWOR
=> D )buf );

            if( lstrcmp( buf, recog->prompt() ) )
                correct = i;
        }

        if( correct == -1 )
            recog->beep( "bad.wav", SND_TOGGLEMIC );

        itoa( recog->confidence(), buf, 10 );

        SetWindowText( GetDlgItem( hwnd, CONFIDENCE_TEXT_ID ), buf );

        recog->nextUtt();

        recog->nextPrompt();

        setupPrompt( hwnd );

        static bool backupDisabled = TRUE;

        if( backupDisabled )
        {
            EnableWindow( GetDlgItem( hwnd, BACKUP_BTN_ID ), TRUE );
            backupDisabled = 0;
        }
      }
      else
      {

          recog->beep( "bad.wav" );

          recog->killUtt();
      }

      if( recog->peekUtt() )
          postSpeechEvent( ch, SD_CHANNEL_START );
      else
      {
          MSG msg;

          while( PeekMessage( &msg, hWnd, UM_CHANNELSTART, UM_CHANNELSTART, PM
 => _REMOVE | PM_NOYIELD ) )
                ;
      }

    }
}


void TrainDlgWin::onDestroy( HWND )
{
    if( speechTask )
    {
        if( channel )
        {
            delete channel;
            channel = 0;
        }
```

```
        if( recog )
        {
            delete recog;
            recog = 0;
        }

        delete speechTask;
        speechTask = 0;
    }

    PostQuitMessage( 0 );
}


long FAR PASCAL _export TrainDlgWin::wndProc( HWND hWnd, UINT message, WPARAM wP
  => aram, LPARAM lParam )
{
    messageHandled = TRUE;

    long returnValue = 0;

    switch( message )
    {
        handleMessage( WM_INITDIALOG, hWnd, wParam, lParam, onInitDialog );
        handleMessage( WM_DESTROY, hWnd, wParam, lParam, onDestroy );
        handleMessage( WM_COMMAND, hWnd, wParam, lParam, onCommand );
        handleMessage( UM_CHANNELSTART, hWnd, wParam, lParam, onSpeech );
    }

    return returnValue;
}



/////////////////////////////////////////////////////////////////////
// AppWin
/////////////////////////////////////////////////////////////////////

class AppWin
{
 public:
    static long FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );

    static HWND  hWnd;
    static HTASK hTask;

 protected:

    static bool messageHandled;

    // wndProc() services
    static bool onCreate( HWND hWnd, CREATESTRUCT FAR* lpCreateStruct );
    static void onDestroy( HWND hWnd );
    static void onNCDestroy( HWND hWnd );
    static void onClose( HWND hWnd );
    static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y );

    friend int PASCAL WinMain( HINSTANCE, HINSTANCE, LPSTR, int );
};


// Static Global Variables of AppWin
HWND            AppWin::hWnd;
HTASK           AppWin::hTask;
bool            AppWin::messageHandled;
```

```
void TrainDlgWin::onCommand( HWND hWnd, UINT cmd, HWND, UINT )
{
    switch ( cmd )
    {
     case CHOICE_LIST_ID:
        break;

     case USR_COMBO_ID:
        break;

     case SAVEEXIT_BTN_ID:
        if( recog && channel && recog->getUser() )
        {
            channel->flush();
            recog->saveUser();
        }

    case EXIT_BTN_ID:
       EndDialog( hWnd, 0 );
       break;

    case BACKUP_BTN_ID:
       {
           int maxBack = (recog->promptIndex() % 3);

           if( maxBack == 0 ) maxBack = 3;

           if( recog->getChannel()->canBackup( maxBack ) )
           {
               int wasListening = recog->isListening();

               if( wasListening )
                   recog->listen( FALSE );

               recog->notifyChannel( 0, SD_CHANNEL_BACKUP );

               recog->nextUtt();

               recog->nextPrompt();

               if( wasListening )
                   recog->listen( TRUE );

               setupPrompt( hWnd );
           }
       }
       break;

   case MICROPHONE_BTN_ID:
       if( recog == 0 )
       {
           char promptName[ LINE_SIZE ];

           if( !GetDlgItemText( hWnd, PROMPT_COMBO_ID, promptName, LINE_SIZE )
=> )
               return;

           switch( usrName[ strlen( usrName ) - 1 ] )
           {
            case 'm':
            case 'M':
               digitsVoc = malDigitsVoc;
               cityVoc = malCityVoc;
                       WENROL.CPP  7-22-95  11:36a
```

```
               break;
            case 'f':
            case 'F':
               digitsVoc = femDigitsVoc;
               cityVoc = femCityVoc;
               break;

            default:
               return;
           }

           HCURSOR hCursor = SetCursor( LoadCursor( 0, IDC_WAIT ) );

           channel = new WindowsAdaptChannel( promptName );

           recog = new Recognizer( channel );

           if( recog->setUser( usrName )  == 0 ||
               recog->setVoc( digitsVoc ) == 0 ||
               recog->setVoc( cityVoc )   == 0 )
           {
               MessageBox( hWnd, "Could not open either digits.voc or ctall.voc
=>  or user file!",
                               0, MB_ICONEXCLAMATION | MB_OK );

               EndDialog( hWnd, 0 );
           }

           channel->open( postSpeechEvent );

           SetWindowText( GetDlgItem( hWnd, MICROPHONE_BTN_ID ), "Microphone Of
=> f" );

           SetCursor( hCursor );
       }
       else
       {
           recog->listen( !recog->isListening() );

           SetWindowText( GetDlgItem( hWnd, MICROPHONE_BTN_ID ),
                          recog->isListening() ? "Microphone On" : "Microphone
=> Off" );

           if( recog->isListening() && recog->prompt() == 0 )
               recog->nextPrompt();

           setupPrompt( hWnd );

       }
       break;

   case WORD_PAUSE_SLIDE_ID:
       {
           wordPause = GetScrollPos( GetDlgItem( hWnd, WORD_PAUSE_SLIDE_ID ), S
=> B_CTL );

           if( recog )
           {
               bool wasListening = recog->isListening();

               if( wasListening )
                   recog->listen( 0 );

               speechTask->setPar( "word-pause", (short )wordPause );

               if( wasListening )
                          Page 6 of 8
```

```
                recog->listen( 1 );
            }
        }
        break;
    }
}


int PASCAL WinMain( HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR lpszCmdLine,
  => int /* nCmdShow */ )
{
    static const char name[] = "DragCpp";

    SetMessageQueue( 128 );

    if( !hPrevInstance )
    {
        WNDCLASS wndclass;

        wndclass.style           = CS_HREDRAW | CS_VREDRAW;
        wndclass.lpfnWndProc     = AppWin::wndProc;
        wndclass.cbClsExtra      = 0;
        wndclass.cbWndExtra      = 0;
        wndclass.hInstance       = hInst;
        wndclass.hIcon           = LoadIcon( 0, MAKEINTRESOURCE( IDI_MY_ICON ) );
        wndclass.hCursor         = LoadCursor( 0, IDC_ARROW );
        // wndclass.hIcon          = LoadIcon( hInst, "merc.ico" );
        // wndclass.hCursor        = LoadCursor( 0, IDC_ARROW );
        wndclass.hbrBackground = 0;
        wndclass.lpszMenuName  = 0;
        wndclass.lpszClassName = name;

        RegisterClass( &wndclass );
    }

    ::hInstance = hInst;        // the global hInstance for the sliderctl
    slideControl.libMainInit(); // register the slider class

    HWND hWnd = CreateWindow( name,
                              name,
                              WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
                              CW_USEDEFAULT,
                              CW_USEDEFAULT,
                              CW_USEDEFAULT,
                              CW_USEDEFAULT,
                              0,
                              0,
                              hInst,
                              lpszCmdLine );

    assert( hWnd == AppWin::hWnd );

    ShowWindow( AppWin::hWnd, SW_HIDE );
//    ShowWindow( hWnd, nCmdShow );
//    UpdateWindow( hWnd );

    MSG msg;

    while( GetMessage( &msg, 0, 0, 0 ) )
    {
        TranslateMessage( &msg );
        DispatchMessage( &msg );
    }
```

MEMROL.CPP 7-22-95 11:36a

```
    return msg.wParam;
}


void AppWin::onCommand( HWND hWnd, UINT cmd, HWND, UINT )
{
    switch ( cmd )
    {
     case WM_INIT_DIALOG:
        {
            // create dialog box proc instances for use later

            FARPROC lpfn = MakeProcInstance( (FARPROC )EnrollDlgWin::wndProc, hI
  => nstance );

            if( DialogBox( hInstance, "EnrollDialog", hWnd, (DLGPROC )lpfn ) )
            {
                FreeProcInstance( lpfn );
                lpfn = MakeProcInstance( (FARPROC )TrainDlgWin::wndProc, hInstan
  => ce );
                DialogBox( hInstance, "TestDialog", hWnd, (DLGPROC )lpfn );
            }

            FreeProcInstance( lpfn );

            PostMessage( hWnd, WM_CLOSE, 0, 0 );
        }
        break;


     default:
        messageHandled = FALSE;
    }
}


bool AppWin::onCreate( HWND hwnd, CREATESTRUCT FAR* )
{
    hTask = GetCurrentTask();

    hWnd = hwnd;

    PostMessage( hwnd, WM_COMMAND, WM_INIT_DIALOG, 0 );

    return TRUE;
}


void AppWin::onDestroy( HWND )
{
    PostQuitMessage( 0 );
}

void AppWin::onNCDestroy( HWND )
{
}

void AppWin::onClose( HWND hWnd )
{
    DestroyWindow( hWnd );
}
```

Page 7 of 8

```
long FAR PASCAL _export AppWin::wndProc( HWND hWnd, UINT message, WPARAM wParam,
  => LPARAM lParam )
{
    messageHandled = TRUE;

    long returnValue = 0;

    switch( message )
    {
        handleMessage( WM_CREATE, hWnd, wParam, lParam, onCreate );
        handleMessage( WM_DESTROY, hWnd, wParam, lParam, onDestroy );
        handleMessage( WM_CLOSE, hWnd, wParam, lParam, onClose );
        handleMessage( WM_COMMAND, hWnd, wParam, lParam, onCommand );

     default:
        messageHandled = FALSE;

        break;
    }

    return( messageHandled
            ? returnValue
            : DefWindowProc( hWnd, message, wParam, lParam ) );
}
```

WENROL.CPP 7-22-95 11:36a

Page 8 of 8

```
/////////////////////////////////////////////////////////////////////
//
// NOMURA Trading Workstation, speech server application
//
// comm module.  This module is responsible for controlling the serial
//               port.
//
// Original Version:  Hal Strausberg  3/28/94
//
// Modification History:
//
//   Date       Person      Change
//   ----       ------      ------
//
//
// Copyright (C) 1994  Dragon Systems Inc.
//
/////////////////////////////////////////////////////////////////////


#include "deamon.h"

extern "C" {

#include <stdlib.h>
#include <string.h>
#include <assert.h>

}  /* end of extern "C" */

//#include "dragcpp.h"

//----------------- Globals -----------------------------------------

Comm* Comm::com1;
Comm* Comm::com2;
Comm* Comm::com3;
Comm* Comm::com4;

CommPars Comm::defaultPars( 2, 9600, NOPARITY, 8, ONESTOPBIT );


//-------------------------------------------------------
// Workhorse timer callback proc for checking serial ports.
//-------------------------------------------------------
void Comm::timerProcPrimitive()
{
    // check for input from the serial port.
    if( comRead() > 0 )
    {
        curNoChar = 0;

        // send a message to our window to parse the message
        PostMessage( hWnd, UM_COMM_CHECK_MSG, (WORD )0, (DWORD )this );
    }
    else
    {
        curNoChar++;

        if( curNoChar >= NOCHAR_TIMEOUT )
            needResync = TRUE;

        if( outGoing)
        {
            // see if we should send resync message or have a message to send
            comWrite();
```

DEAMON.CPP  7-22-95  10:02a

```
        }

        if( needResync)
        {
            curNoChar = 0;
            needResync = FALSE;
            PostMessage( hWnd, UM_COMM_SEND_SYNC_MSG, 0, (DWORD )this );
        }
    }
    return;
}


void CALLBACK _export Comm::com1TimerProc( HWND, UINT, UINT, DWORD )
{
    if( com1 )          com1->timerProcPrimitive();
}
void CALLBACK _export Comm::com2TimerProc( HWND, UINT, UINT, DWORD )
{
    if( com2 )          com2->timerProcPrimitive();
}
void CALLBACK _export Comm::com3TimerProc( HWND, UINT, UINT, DWORD )
{
    if( com3 )          com3->timerProcPrimitive();
}
void CALLBACK _export Comm::com4TimerProc( HWND, UINT, UINT, DWORD )
{
    if( com4 )          com4->timerProcPrimitive();
}


//-------------------------------------------------------
// constructor
//-------------------------------------------------------

Comm::Comm( HWND hwnd, CommPars* defPars )
{
    if( defPars == 0 )
        defPars = &defaultPars;

    switch( defPars->port )
    {
     case 1:            global = &com1;         break;
     case 2:            global = &com2;         break;
     case 3:            global = &com3;         break;
     case 4:            global = &com4;         break;

     default:           global = 0;
        err = IE_OPEN;
        return;
    }

    if( *global != 0 )
    {
        err = IE_OPEN;
        return;
    }

    params.port     = defPars->port;
    params.baud     = defPars->baud;
    params.parity   = defPars->parity;
    params.databits = defPars->databits;
    params.stopbits = defPars->stopbits;
    params.breaklen = defPars->breaklen;
    params.inx      = defPars->inx;
    params.outx     = defPars->outx;
```

Page 1 of 5

```
    params.cts      = defPars->cts;
    params.dsr      = defPars->dsr;
    params.dtr      = defPars->dtr;
    params.rts      = defPars->rts;

    hWnd = hwnd;
    index = 0;
    curNoChar=0;
    needResync=TRUE;
    outGoing=0;

    if( comConnect() )
        *global = this;
}

//--------------------------------------------------
// destructor
//--------------------------------------------------
Comm::~Comm()
{
    comDisconnect();

    // kill the timer if started.
    if(  wTimerID )
        KillTimer( NULL, wTimerID );
}

//--------------------------------------------------
// connect
//--------------------------------------------------
BOOL Comm::comConnect()
{
    /* see if already open */
    if( connectId >= 0 )
    {
        err = IE_OPEN;
        return FALSE;
    }

    /* open the connection and establish the settings - show hourglass */
    err = 0;

    char name[20] = "COM";

    itoa( params.port, name + 3, 10 );    name[ 4 ] = '\0';

    if( (connectId = OpenComm( name, INBUFSIZE, OUTBUFSIZE )) >= 0 )
    {
        DCB dcb;
        comGetWindowsParams( &params, &dcb);
        dcb.Id = (BYTE)params.port-1;
        err = SetCommState( &dcb );
    }
    else
        err = connectId;

    BOOL rtn = (err == 0);

    if( rtn == 1 )
    {
        void CALLBACK _export ( *tmProc )( HWND, UINT, UINT, DWORD );

        tmProc = 0;

        switch( params.port )
```

DEAMON.CPP 7-22-95 10:02a

```
        {
         case 1:  global=&com1; *global=0;  tmProc=com1TimerProc;    break;
         case 2:  global=&com2; *global=0;  tmProc=com2TimerProc;    break;
         case 3:  global=&com3; *global=0;  tmProc=com3TimerProc;    break;
         case 4:  global=&com4; *global=0;  tmProc=com4TimerProc;    break;

         default:
            global = 0;
            err = IE_OPEN;
            rtn = 0;
        }

        // GREG --- I am turning off the polling for mercury
        // start a timer to read the serial port.

        // wTimerID = ( tmProc != 0
        //           ? SetTimer( NULL, TIMER_EVENT, TIMER_INTERVAL, tmProc );
        //           : 0 );
        //
        // rtn = (wTimerID != 0);
    }

    /* close again if settings were NFG */
    if( rtn )
        *global = this;

    else if( connectId >= 0 )
    {
        CloseComm( connectId );

        connectId = NFG;

        global = 0;
    }

    else
        global = 0;

    return rtn; /* TRUE if OK */
}

//--------------------------------------------------
// disconnect from the device
//--------------------------------------------------
BOOL Comm::comDisconnect()
{
    err = 0;

    /* close the connection */
    if( global && *global==this )
        *global = 0;

    global = 0;

    if( connectId >= 0)
    {
        /* close a Windows connection */
        err = CloseComm( connectId );

        connectId = NFG;
    }

    return( err == 0 );
}

//--------------------------------------------------
```

Page 2 of 5

```
// reset the device
//.........................................................
BOOL Comm::comReset()
{
    int rtn;
    DCB dcb;

    /* close the connection */
    err = -1;

    if( connectId >= 0 )
    {
        comGetWindowsParams( &params, (DCB far *)&dcb);
        dcb.Id = (BYTE)params.port-1;

        err = SetCommState((DCB far *)&dcb);

        err |= EscapeCommFunction( connectId, RESETDEV );

        /* set device to ready */
        err |= EscapeCommFunction( connectId, SETDTR );
        err |= EscapeCommFunction( connectId, SETXON );
        err |= EscapeCommFunction( connectId, CLRRTS );
    }

    return( err == 0 );
}


//.........................................................
// read a string of chars from the comm device
//.........................................................
int Comm::comRead()
{
    int count;

    err = 0;

    if( connectId < 0 )          return 0;

    /* always try to read */
    count = ReadComm( connectId, &xrBuffer[ index ], INBUFSIZE-index );

    if( count <= 0 )
    {
        /* check for errors */
        err = GetCommError( connectId, NULL ) & 0x1FF;

        GetCommEventMask( connectId, 0xFFFF );

        if( err == CE_FRAME )
            err = 0;

        else if( err > 0 )
            err += 1024;

        count = -count;
    }

    return index += count;
}

//.........................................................
// write a string of chars to the comm device
//.........................................................
int Comm::comWrite()
```

DEAMON.CPP 7-22-95 10:02a

```
{
    int   count;

    if( connectId >= 0 )
    {
        /* here we'll do an output regardless of bytes still being queued */
        if( (count = comCheckOutputStatus()) >= 0 )
            count = comDoOutput( xBuffer, lstrlen( xBuffer ) );
    }
    return count;
}


//.........................................................
int Comm::comWrite( LPSTR lpStr, int len )
{
    int   count;

    if( connectId >= 0 )
    {
        /* here we'll do an output regardless of bytes still being queued */
        if( (count = comCheckOutputStatus() )>= 0)
            count = comDoOutput( lpStr, len );
    }
    return count;
}

//.........................................................
// Processes the incoming message.
//.........................................................
BOOL Comm::checkMessage()
{
    // we are throwing away the message...
    // if you want to do something... inherit from and overload CheckMessage()
    index = 0 ;
    xrBuffer[ 0 ] = '\0';           // zero'ed out recieved message

    return 1;
}


//.........................................................
// does the right thing with an incoming sync message.
//.........................................................
BOOL Comm::sendSyncMsg()
{
    return 1;
}



//.........................................................
// send a break sequence to the device
//.........................................................
void Comm::comSendBreak( int len )
{
    DWORD ticks;
    int rtn;

    if( connectId >= 0 )
    {
        if( !len )
            len = params.breaklen;

        rtn = SetCommBreak( connectId );
```

Page 3 of 5

```
        if( !rtn )
        {
            /* break is on */
            ticks = GetTickCount();

            while( GetTickCount() < ticks + len)
                ;

            ClearCommBreak( connectId );
        }
    }
}

//--------------------------------------------------
// read and discard input until line is clear or timeout occurrs
//--------------------------------------------------
BOOL Comm::comClearLine( DWORD dwQuiet, DWORD dwMax )
{
    DWORD tNow, tStart, tInput;
    char str[ 128 ];

    tStart = tInput = GetTickCount();

    for(;;)
    {
        tNow = GetTickCount();

        if( tNow > tStart + dwMax )
            return FALSE;

        if( tNow > tInput + dwQuiet )
            return TRUE;

        if( comRead( str, 128) )
            tInput = tNow;
    }
}

//--------------------------------------------------
// return current com parameters
//--------------------------------------------------
int Comm::comGetParams( CommPars* lpParams )
{
    *lpParams = params;

    return connectId;
}

//--------------------------------------------------
// set com parameters
//--------------------------------------------------
BOOL Comm::comSetParams( CommPars* lpParams )
{
    if( connectId >= 0 )
    {
        DCB dcb;

        err = 0;

        comGetWindowsParams( lpParams, &dcb );

        dcb.Id = (BYTE)params.port-1;

        err = SetCommState( &dcb );
    }
```

DEAMON.CPP 7-22-95 10:02a

```
    if( err == 0 )
    {
        params = *lpParams;

        return 1;
    }

    return 0;
}

//--------------------------------------------------
// return rx and tx queue counts, or flush the queues
//--------------------------------------------------
void Comm::comGetQueueCounts( LPINT lpRxq, LPINT lpTxq )
{
    COMSTAT comstat;

    /* using Windows */
    GetCommError( connectId, &comstat );

    *lpRxq = comstat.cbInQue;

    *lpTxq = comstat.cbOutQue;

    return;
}

//--------------------------------------------------
// convert our params struct to a Windows DCB struct
//--------------------------------------------------
void Comm::comGetWindowsParams( CommPars* lpPrms, DCB far *lpDCB )
{
    GetCommState( connectId, lpDCB );

    lpDCB->BaudRate = lpPrms->baud;
    lpDCB->ByteSize = lpPrms->databits;
    lpDCB->Parity = lpPrms->parity;
    lpDCB->StopBits = lpPrms->stopbits;
    lpDCB->RlsTimeout = 0;
    lpDCB->CtsTimeout = 0;
    lpDCB->DsrTimeout = 0;
    lpDCB->fBinary = TRUE;
    lpDCB->fParity = 0;
    lpDCB->fOutxCtsFlow = (BYTE)lpPrms->cts;
    lpDCB->fOutxDsrFlow = (BYTE)lpPrms->dsr;
    lpDCB->fDtrDisable = 0;
    lpDCB->fOutX = (BYTE)lpPrms->outx;
    lpDCB->fInX = (BYTE)lpPrms->inx;
    lpDCB->fPeChar = 0;
    lpDCB->fNull = 0;
    lpDCB->fChEvt = 0;
    lpDCB->fDtrflow = (BYTE)lpPrms->dtr;
    lpDCB->fRtsflow = (BYTE)lpPrms->rts;
    lpDCB->XonChar = XON;
    lpDCB->XoffChar = XOFF;
    lpDCB->XonLim = INBUFSIZE / 3 * 2 + 1;
    lpDCB->XoffLim = INBUFSIZE / 3 * 2 + 1; /* two thirds of the buffer size */
    lpDCB->PeChar = 0;
    lpDCB->EofChar = 0;
    lpDCB->EvtChar = 0;
    lpDCB->TxDelay = 0;

    return;
}
```

Page 4 of 5

```
//··················································
// find out if an output error has occurred and
//   how many output bytes are still queued
//··················································
int Comm::comCheckOutputStatus()
{
    COMSTAT comstat;            /* Windows status struct */

    /* we're using Windows */
    err = ~(CE_FRAME | CE_BREAK) & GetCommError( connectId, &comstat );

    if( err != 0 )
        return -1;

    return comstat.cbOutQue;    /* number of bytes remaining to be sent */
}

//··················································
// performs write function for appropriate board/port, assuming that
//   the output status has already been checked and is ok
//··················································
int Comm::comDoOutput( LPSTR lpStr, int len )
{
    int count;

    /* we're using Windows */
    /* WriteComm will return immediately, having written fewer
    than <len> bytes if needed */
    count = WriteComm( connectId, lpStr, len );

    if( count < 0 )
    {
        err = GetCommError( connectId, NULL )& 0x1FF;

        //   event = GetCommEventMask(connectId, 0xFFFF);
        GetCommEventMask( connectId, 0xFFFF );
    }

    // EscapeCommFunction( connectId, SETXON );
    return count;
}

//··················································
// flush the input and output buffers
//··················································
void Comm::comFlushQueues()
{
    FlushComm( connectId, 0);
    FlushComm( connectId, 1);
}

//···················· E O F ·························
```

DEAMON.CPP 7-22-95 10:02a

Page 5 of 5

```
/**************************************************************************
        Description

        DRAGCPP.CPP

        Wrapper around Dragon Systems' VoiceTools 1.1(TM), toolkit
        to expose the Speech Driver API (SDAPI) to customers. The wrapper
        simplifies some of the tasks to set up a recognizer.

    Copyright (c) 1993-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1993 - 1995

**************************************************************************/



extern "C" {

#include "std.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <bios.h>
#include <limits.h>
#include "liberr.h"

}  /* end of extern "C" */

#include <mmsystem.h>
#include "dragcpp.h"

#define DEBUG 1


// doubly linked list
Deq< VocStateMachine > VocStateMachine::vscDeq;
Deq< UserPhonetics > UserPhonetics::upDeq;


// Initialize activeUser to 0
SD_USER UserPhonetics::activeUser = 0;

// Beep function used in itsGoodAdapt() or itsBad() in WAPP.cpp
void beep( char* wavFileName, int mode )
{
    sndPlaySound( wavFileName, mode & 0x1f );
}

/////////////////////////////////////////////////////////////////////////
// SpeechTask
/////////////////////////////////////////////////////////////////////////
char SpeechTask::versbuf[128] = { '\0' };

// constructor
// parameters:  Task name
//                              Error handling function CALLBACK

speechTask:SpeechTask( char *taskName, void SD_CALLBACK (*sdErrHandler)(int, cha
  => r*) )
{
    assert( taskName );
    if( !versbuf[0] ) {
        SDError_SetHandler( sdErrHandler );
        SDApi_Init( );
        SDApi_GetVersion( versbuf, sizeof( versbuf ) );
        //    printf( "%s\n", versbuf );
        //    printf( "Copyright ( c ) Dragon Systems, Inc. 1991, 1992\n" );
    }

    // If our task is already opened, close it!  We don't use any state from
    // a previous execution.

    task = SDTask_GetHandle( taskName );
    if( task ) SDTask_Close( task );

    // open the task
    task = SDTask_New( taskName );

    assert( task );

    // because there can be multiple instance of this one program, the task
    // name is constructed using the instance handle in the WinMain routine
    // so that every instance will be unique

}

/////////////////////////////////////////////////////////////////////////
// SpeechTask
/////////////////////////////////////////////////////////////////////////

SpeechTask::~SpeechTask()
{
    SDTask_Close( task );
}

// Get Parameters: 1 byte value
void SpeechTask::getPar( const char* parName, char* val )
{
    SDPar_GetValue( SDPar_GetHandle( parName ), val, MAX_TEXT_PAR );
#ifdef DEBUG
    //    printf( "%s = %s\n", parName, val );
#endif
}

// Get Parameters: 2 byte value
void SpeechTask::getPar( const char* parName, int16* val )
{
    SDPar_GetValue( SDPar_GetHandle( parName ), val, sizeof(int16) );
#ifdef DEBUG
    //    printf( "%s = %hd\n", parName, *val );
#endif
}

// Get Parameters: 4 byte value
void SpeechTask::getPar( const char* parName, int32* val )
{
    SDPar_GetValue( SDPar_GetHandle( parName ), val, sizeof(int32) );
#ifdef DEBUG
    //    printf( "%s = %ld\n", parName, *val );
#endif
}

// Get Parameters: 4 byte value, unsigned
void SpeechTask::getPar( const char* parName, uns32* val )
{
    SDPar_GetValue( SDPar_GetHandle( parName ), val, sizeof(uns32) );
#ifdef DEBUG
    //    printf( "%s = %lu\n", parName, *val );
#endif
```

DRAGCPP.CPP 7-22-95 11:35a

Page 1 of 17

```
}

// Set Parameters: 1 byte value
void SpeechTask::setPar( const char* parName, char* val )
{
    SDPar_SetValue( SDPar_GetHandle( parName ), (void *) val, strlen(val) +1 );
#ifdef DEBUG
    getPar( parName, val );
#endif
}

// Set Parameters: 2 byte value

void SpeechTask::setPar( const char* parName, int16 val )
{
    SDPar_SetValue( SDPar_GetHandle( parName ), (void *) &val, sizeof( int16 ) )
  » ;
#ifdef DEBUG
    getPar( parName, &val );
#endif
}

// Set Parameters: 4 byte value
void SpeechTask::setPar( const char* parName, int32 val )
{
    SDPar_SetValue( SDPar_GetHandle( parName ), (void *) &val, sizeof( int32 ) )
  » ;
#ifdef DEBUG
    getPar( parName, &val );
#endif
}

// Set Parameters: 4 byte value, unsigned
void SpeechTask::setPar( const char* parName, uns32 val )
{
    SDPar_SetValue( SDPar_GetHandle( parName ), (void *) &val, sizeof( uns32 ) )
  » ;
#ifdef DEBUG
    getPar( parName, &val );
#endif
}



// UttChannel constructor
UttChannel::UttChannel()
{
    uttUtt=0; uttIndex = 0;
    uttPrompt = new char[ UTT_PROMPT_LENGTH ];
    uttPrompt[ 0 ] = '\0';
    choices = new ArrayOfRecogResults;
}

UttChannel::~UttChannel()
{
    if( utt() )         killUtt();
    if( uttPrompt ) {    delete [] uttPrompt;    uttPrompt=0; }
    if( choices ) {    delete choices;         choices = 0; }
}

int UttChannel::open( SD_CHANNEL_EVENT_HANDLER ) { return( utt() != 0 ); }

void UttChannel::close() { }

SD_UTT UttChannel::startUtt( ) { return utt(); }
```

```
SD_UTT UttChannel::peekUtt( ) { return 0; }


void UttChannel::setPrompt( const char* newPrompt )
{
    if( uttPrompt == 0 )
        uttPrompt = new char[ UTT_PROMPT_LENGTH ];

    strncpy( uttPrompt, newPrompt, UTT_PROMPT_LENGTH );
}

// Get the ith  results from choice list
const RecogResult* UttChannel::result( int ithResult )
{
    if( ithResult >= resultCount() ) return 0;

    RECOG_RESULT_ENTRY *presult = &(choices->recogResults[ 0 ]);


    for( int i=0; i < ithResult; i++ ) {
        presult = ( RECOG_RESULT_ENTRY * )
        ( ( char * )presult + sizeof( RECOG_RESULT_ENTRY ) +
          sizeof( SD_WORD_SPEC )*( presult->nWords - 1 ) );
    }
    return (const RecogResult* )presult;
}

// Put name of recognized utterance in buf
int UttChannel::resultName( char* buf, int sizebuf, int ithResult )
{
    const RecogResult *res = result( ithResult );
    if( res )
        return res->getName( buf, sizebuf );
    else if( sizebuf && buf )
        buf[0] = '\0';

    return 0;
}

// Put spaces in between digits
void UttChannel::seperateDigits( int digitLen )
{
    char *s = uttPrompt;

    assert( s );

    assert( digitLen > 0 );

    char *newPrompt = new char[ UTT_PROMPT_LENGTH ];

    char *ss = newPrompt;

    while( *s )
    {
        while( isspace( *s ) )
            ++s;

        if( isdigit( *s ) )
        {
            if( digitLen-- == 0 )
                break;

            *ss++ = *s++;
            *ss++ = ' ';
        }
        else
```

```
            break;
        }
        if( ss != newPrompt )
            --ss;

        *ss = '\0';

        delete [] uttPrompt;

        uttPrompt = newPrompt;
}


bool UttChannel::isListening()    { return( utt() != 0 ); }
bool UttChannel::listen( bool )   { return !isListening(); }
bool UttChannel::notifyChannel( SD_CHANNEL, SD_CHANNEL_EVENT ) { return isListen
  => ing(); }

// delete utterance from the channel
void UttChannel::killUtt()
{
    if( utt() ) {       SDUtt_Delete( utt() );      uttUtt  = 0;        }
}

// Get next utterance
void UttChannel::nextUtt()
{
    if( utt() ) {       SDUtt_Delete( utt()  );     uttUtt  = 0;        }
}

void UttChannel::purge() { }    // UttChannel is not buffered...
void UttChannel::flush() { }

// Move utterance in channel out of it.
SD_UTT UttChannel::giveAwayUtt( UttChannel* toMe )
{
    assert( toMe );

    char *tmp=toMe->uttPrompt;
    toMe->uttPrompt = uttPrompt;
    uttPrompt = tmp;

    if( uttPrompt )
        uttPrompt[0] = '\0';

    toMe->uttIndex = uttIndex;

    ArrayOfRecogResults *tmpChoices = toMe->choices;
    toMe->choices = choices;
    choices = tmpChoices;

    choices->recogVoc = 0;
    choices->recogUser = 0;

    toMe->killUtt();

    SD_UTT rtn = toMe->uttUtt = utt();
    uttUtt  = 0;

    assert( rtn < 20 );

    return rtn;
}

SD_UTT UttChannel::giveAwayUtt()
{
     SD_UTT rtn = utt();
     uttUtt = 0;
     if( prompt() )
         *uttPrompt = '\0';

     choices->recogVoc = 0;
     choices->recogUser = 0;

     return rtn;
}


// get next prompt
bool UttChannel::peekPrompt( char* buf, int bufSize, int promptsAhead )
{
    if( promptsAhead )
        return 0;
    else
        strncpy( buf, prompt(), bufSize );

    return 1;
}


bool UttChannel::canBackup( int n )
{
    return !n;
}

bool UttChannel::nextPrompt()
{
    return( uttPrompt && *uttPrompt );
}


// beep
void UttChannel::beep( char* wavFileName, int mode )
{
    ::beep( wavFileName, mode );
}

// WindowsLiveChannel constructor

WindowsLiveChannel::WindowsLiveChannel( const char* promptfname )
{
    mic=0;    uttUtt=0;    flag=0;    uttPrompt=0;  promptFile=0;

    uttIndex = -1;      nextIndex = 0;          chanIsOn = 0;

    if( promptfname && *promptfname )
    {
        promptFilename = new char[ strlen(promptfname) +1 ];

        if( !stricmp( stpcpy( promptFilename, promptfname ) -4, ".spc" ) )
            flag |= SCP_PROMPT;
    }
    else
        promptFilename = 0;

}

// destructor
WindowsLiveChannel::~WindowsLiveChannel()
{
    WindowsLiveChannel::close();
    if( promptFilename )
    {
        delete promptFilename;

        promptFilename = 0;
    }
}

int WindowsLiveChannel::open( SD_CHANNEL_EVENT_HANDLER notifyRoutine )
{
    assert( mic==0 );

    // the first channel is the microphone...
    SD_CHANNEL_ITERATOR ctrl;
    SDChannel_Iterate( &ctrl );
    mic = SDChannel_Next( &ctrl );

    if( mic == 0 )
        return 0;

    if( promptFilename && !promptFile ) {
        promptFile = fopen( promptFilename, "r" );
        assert( promptFile );
    }

    SDChannel_Open( mic );
    SDChannel_Claim( mic, YES );

    chanIsOn = 0;

    SDChannel_SetEventHandler( mic, SD_CHANNEL_START, notifyRoutine );

    return 1;
}

void WindowsLiveChannel::close( )
{
    if( mic )
    {
        if( isListening() )
            listen( FALSE );
        SDChannel_Release( mic );
        SDChannel_Close( mic );
        mic=0;
    }
    if( promptFile )
    {
        fclose( promptFile );
        promptFile = 0;
    }
}

bool WindowsLiveChannel::isListening() { return chanIsOn; }

// Make channel listen
bool WindowsLiveChannel::listen( bool turnOn )
{
    assert( mic );

        // if channel is already on, return immediately, otherwise
    return( ( chanIsOn=turnOn ) != 0
            ? SDChannel_SetMicOn( mic, YES, NO )
            : SDChannel_SetMicOff( mic ) );
}
```

```
// Tell channel to start
bool WindowsLiveChannel::notifyChannel( SD_CHANNEL ch, SD_CHANNEL_EVENT ev )
{
    if( ev == SD_CHANNEL_START )
    {
        if( isListening() )
        {
            assert( mic == ch );
            assert( choices );

            choices->recogUser = 0;
            choices->recogVoc = 0;

            return( (uttUtt=SDChannel_Read( ch, 0 )) != 0 );
        }
        else
        {
            SD_UTT u = SDChannel_Read( ch, 0 );

            if( u )
                SDUtt_Delete( u );

            return FALSE;
        }
    } // if SD_CHANNEL_START

    return FALSE;
}

// Get next prompt
bool WindowsLiveChannel::nextPrompt()
{
    if( promptFile == 0 ) return 0;

    if( uttPrompt == 0 )
    {
        uttPrompt = new char[ UTT_PROMPT_LENGTH ];

        uttPrompt[0] = '\0';
    }

    assert( nextIndex >= 0 );
    //    assert( nextIndex >= uttIndex );

    int  c, size = UTT_PROMPT_LENGTH -2;
    char *p;

    for(;;)
    {
        size = UTT_PROMPT_LENGTH -2;
        p = uttPrompt;

        while( (c=getc(promptFile))==' ' || c=='\t' );
        ungetc( c, promptFile );

        while( (c=getc(promptFile)) != EOF  && c != '\n' ) {
            if( size ) {
                size--;
                if( c==' ' || c=='\t' ) {
                    while( (c=getc(promptFile))==' ' || c=='\t' );
                    if( c!='\n' && c!=EOF )    *p++ = ' ';
                    ungetc( c, promptFile );
                }
                else *p++ = c;
            }
        }
        if( UTT_PROMPT_LENGTH -2 != size ) {
            *p = '\0';
            if( flag & SCP_PROMPT ) {
                // need to strip off the last number
                while( --p > uttPrompt && (*p!=' ' || *p!='\t') );
                assert( *p==' ' || *p=='\t' );
                *p++ = '\0';
                uttIndex = atoi( p );
            }
            else
                uttIndex = nextIndex;

            break;
        }
        else if( c==EOF ) {
            uttPrompt[0] = '\0';;
            uttIndex = -1;
            return 0;
        }
    }

    return( uttPrompt && *uttPrompt );
}

// peek for an utterance
SD_UTT WindowsLiveChannel::peekUtt( )
{
    return( chanIsOn ? SDChannel_Read( mic, CH_PEEK ) : 0 );
}

// Get next channel
void WindowsLiveChannel::nextUtt()
{
    if( utt()  )
    {
        SDUtt_Delete( utt()  );
        uttUtt  = 0;
        ++nextIndex;
    }
}

// beep
void WindowsLiveChannel::beep( char* wavFileName, int mode )
{
    if( (mode & SND_TOGGLEMIC) && isListening() )
    {
        listen( FALSE );
        ::beep( wavFileName, mode & ~SND_ASYNC );
        listen( TRUE );
    }
    else
        ::beep( wavFileName, mode );
}


// WindowsFileChannel constructor
// Initialize prompt file name and utt file name
WindowsFileChannel::WindowsFileChannel( const char* uttFname, const char* prompt
  => Fname )
: WindowsLiveChannel( promptFname ) {

    notifyRoutine = 0;
    uttFile = 0;

    if( uttFname && *uttFname )
```

DRAGCPP.CPP 7-22-95 11:35a

```
    {
        uttFilename = new char( strlen(uttFname) +1 );

        strcpy( uttFilename, uttFname );
    }
    else
        uttFilename = 0;
}


WindowsFileChannel::~WindowsFileChannel()
{
    WindowsFileChannel::close();
}

// Open the channel
int WindowsFileChannel::open( SD_CHANNEL_EVENT_HANDLER notifyMe )
{
    assert( uttFilename && *uttFilename );

        // open list file
    if( promptFilename && !promptFile )
    {
        promptFile = fopen( promptFilename, "r" );

        assert( promptFile );
    }

        // open utt file
    if( uttFilename && !uttFile )
    {
        uttFile = SDChannel_NewFile( uttFilename, "r" );
        assert( uttFile );

        // open channel
        SDChannel_Open( uttFile );
        SDChannel_Claim( uttFile, TRUE );
    }

        // if there is no utterance, close the file channel
    if( !uttFile )
    {
        WindowsFileChannel::close();

        assert( uttFile != 0 );
    }

    assert( notifyMe );

    notifyRoutine = notifyMe;

    return 1;
}

// close the file channel
void WindowsFileChannel::close()
{
    if( uttFile )
    {
        SDChannel_Close( uttFile );    uttFile = 0;
    }
    WindowsLiveChannel::close();
}

// Start utt, seek the beginning, and read
// Function called from notifyChannel
```

Page 5 of 17

```
SD_UTT WindowsFileChannel::startUtt( )
{
    if( utt() )
        return utt();

    if( uttIndex < 0 )
        return 0;

    SDChannel_Seek( uttFile, uttIndex, SM_BEGINNING );
    return utUtt = SDChannel_Read( uttFile, CH_WAITSTART|CH_WAITEND );
}


SD_UTT WindowsFileChannel::peekUtt( )
{
    return 0;
}

void WindowsFileChannel::nextUtt()
{
    WindowsLiveChannel::nextUtt();

    notifyRoutine( uttFile, SD_CHANNEL_START );
}

// turn channel on, if you need to.
bool WindowsFileChannel::listen( bool turnOn )
{
    assert( uttFile );

    bool channelWasOn = chanIsOn;

    if( (chanIsOn=turnOn) != 0 )
    {
        ((SD_CHANNEL_EVENT_HANDLER )notifyRoutine)( uttFile, SD_CHANNEL_START );
    }

    return channelWasOn;
}

//
//
bool WindowsFileChannel::notifyChannel( SD_CHANNEL ch, SD_CHANNEL_EVENT ev )
{
    if( ev == SD_CHANNEL_START )
    {
        if( isListening() )
        {
            assert( uttFile == ch );
                // startUtt returns SD_UTT
            return( startUtt() != 0 );
        }
    }
    return FALSE;
}

// delete utterance in the channel
void WindowsFileChannel::killUtt()
{
    if( utt() ) {
        SDUtt_Delete( utt() );    utUtt  = 0;
    }
        // Read the next utterance.
    notifyRoutine( uttFile, SD_CHANNEL_START );
```

DRAGCPP.CPP 7-22-95 11:35a

```
}

////////////////////////////////////////////////////////////////////
// WindowsCollectChannel
////////////////////////////////////////////////////////////////////

WindowsCollectChannel::WindowsCollectChannel( const char* uttFname, const char*
  => promptFname )
: WindowsFileChannel( uttFname, promptFname ) {

    head = point = 0;
    tail = SIZE_UTT_RING -1;
    advance = 1;
}

WindowsCollectChannel::~WindowsCollectChannel()
{
    WindowsCollectChannel::close();
}


SD_UTT WindowsCollectChannel::startUtt()
{
    return utt();
}

//Open channel for collection
int WindowsCollectChannel::open( SD_CHANNEL_EVENT_HANDLER speechEventHandler )
{
    if( uttFilename && !uttFile ) {
        uttFile = SDChannel_NewFile( uttFilename, "w" );
        SDChannel_Open( uttFile );
        SDChannel_Claim( uttFile, TRUE );
    }

    if( !uttFile || !WindowsLiveChannel::open( speechEventHandler ) ) {
        WindowsFileChannel::close();    return 0;
    }

    return 1;
}

// close channel
void WindowsCollectChannel::close()
{
    if( uttFile ) {
        flush();
        SDChannel_Release( uttFile );
        SDChannel_Close( uttFile );    uttFile = 0;
    }
    WindowsLiveChannel::close();
}

// Write utterance to file
void WindowsCollectChannel::outputUtt( UttChannel* uCh )
{
    assert( uCh != 0 );

    if( uCh->utt() != 0 )
    {
        SDChannel_Write( uttFile, uCh->utt() );

        uCh->killUtt();
    }
}
```

Page 6 of 17

```
int WindowsCollectChannel::incHead()
{
    head = incIndex( head );
    ++nextIndex;

    if( head == tail )
    {
        tail = incIndex( tail );

        outputUtt( &(uttRing[ tail ]) );
    }
    return head;
}


SD_UTT WindowsCollectChannel::peekUtt( )
{
    return( chanIsOn ? SDChannel_Read( mic, CH_PEEK ) : 0 );
}

void WindowsCollectChannel::nextUtt()
{
    giveAwayUtt( &(uttRing[ point ]) );

    if( point==head )
        incHead();

    if( !advance )
    {
        int pnt = decIndex( point );

        if( pnt != tail )
            point = pnt;

        advance = 1;
    }
    else if( uttRing[ point ].utt() )
        point = incIndex( point );
}


void WindowsCollectChannel::flush()
{
    if( utt() )
        nextUtt();

    while( (tail = incIndex( tail )) != head )
        outputUtt( &(uttRing[ tail ]) );

    tail = decIndex( tail );

    point = head;
}

void WindowsCollectChannel::purge()
{
    if( utt() )
        killUtt();

    while( (tail = incIndex( tail )) != head )
        uttRing[ tail ].killUtt();

    tail = decIndex( tail );

    point = head;
```

```
}

bool WindowsCollectChannel::canBackup( int numBack )
{
    int space = ( point > tail
                  ? point - tail
                  : SIZE_UTT_RING - tail + point );

    return( space > numBack );
}

bool WindowsCollectChannel::peekPrompt( char* buf, int bufSize, int promptsAhead
  => )
{
    if( promptsAhead == 0 )
    {
        if( prompt() || nextPrompt() )
        {
            strncpy( buf, prompt(), bufSize );
            return TRUE;
        }
        else
        {
            *buf = '\0';
            return FALSE;
        }
    }

    if( promptsAhead > 0 )
    {
        assert( promptsAhead < SIZE_UTT_RING );

        int lastPoint = point;

        bool rtn = TRUE;

        for( int i=0; rtn && i<promptsAhead; ++i )
        {
            if( point==head )
                incHead();

            giveAwayUtt( &(uttRing[ point ]) );

            point = incIndex( point );

            if( point==head )
                rtn &= WindowsLiveChannel::nextPrompt();

            else
                uttRing[ point ].giveAwayUtt( this );
        }

        if( rtn )
        {
            strncpy( buf, prompt(), bufSize );

            if( point==head )
                incHead();

            giveAwayUtt( &(uttRing[ point ]) );

        }
        else
            *buf = '\0';

        point = lastPoint;
        uttRing[ point ].giveAwayUtt( this );

        return rtn;
    }
    else
    {
        int promptsBack = -promptsAhead;

        int space = ( point > tail
                    ? point - tail -1
                    : SIZE_UTT_RING - tail + point -1 );

        if( space < promptsBack )
            return 0;

        int lastPoint = point;

        point -= promptsBack;

        if( point < 0 )
            point += SIZE_UTT_RING;

        strncpy( buf, uttRing[ point ].prompt(), bufSize );

        point = lastPoint;

        return TRUE;
    }
}

bool WindowsCollectChannel::nextPrompt()
{
    if( point==head ) {
        return WindowsLiveChannel::nextPrompt();
    }
    else {
        uttRing[ point ].giveAwayUtt( this );

        killUtt();
    }
    return( uttPrompt && *uttPrompt );
}


bool WindowsCollectChannel::listen( bool turnOn )
{
    assert( mic );

    return( (chanIsOn=turnOn) != 0
            ? SDChannel_SetMicOn( mic, YES, NO )
            : SDChannel_SetMicOff( mic ) );
}


//
bool WindowsCollectChannel::notifyChannel( SD_CHANNEL ch, SD_CHANNEL_EVENT ev )
{
    if( ev == SD_CHANNEL_START )
    {
        if( isListening() )
        {
            assert( mic == ch );

            uttUtt = SDChannel_Read( ch, CH_WAITSTART );
```

DRAGCPP.CPP 7-22-95 11:35a

```
            return( utt() != 0 );
        }
        else
        {
            SD_UTT u = SDChannel_Read( ch, 0 );

            if( u )
                SDUtt_Delete( u );

            return FALSE;
        }
    }
    else if( ev == SD_CHANNEL_BACKUP )
    {
        advance = 0;
    }

    return FALSE;
}

// Delete utterance
void WindowsCollectChannel::killUtt()
{
    if( utt() ) {
        SDUtt_Delete( utt() );    uttUtt  = 0;
    }
}

/////////////////////////////////////////////////////////////////////////
// WindowsAdaptChannel
/////////////////////////////////////////////////////////////////////////

WindowsAdaptChannel::WindowsAdaptChannel( const char* promptFname )
: WindowsCollectChannel( 0, promptFname ) {  recog = 0; }

WindowsAdaptChannel::~WindowsAdaptChannel()
{
    WindowsAdaptChannel::close();

    if( recog )
    {
        delete recog;
        recog = 0;
    }
}

void WindowsAdaptChannel::outputUtt( UttChannel* uCh )
{
    assert( uCh != 0 );

    if( uCh->choices->recogVoc == 0 || uCh->choices->recogUser == 0 ||
        uCh->utt() == 0 )
    {
        uCh->killUtt();
        return;
    }

    if( recog == 0 )
        recog = new Recognizer( uCh );

    else
        recog->setChannel( uCh );

    if( recog->getVoc() != uCh->choices->recogVoc )
        recog->setVoc( uCh->choices->recogVoc );
```

Page 8 of 17

```
    if( recog->getUser() != uCh->choices->recogUser )
        recog->setUser( uCh->choices->recogUser );

    uCh->choices->isCont ? recog->contAdapt() : recog->adapt();

    uCh->killUtt();
}

bool WindowsAdaptChannel::open( SD_CHANNEL_EVENT_HANDLER speechEventHandler )
{
    if( !WindowsLiveChannel::open( speechEventHandler ) )
    {
        WindowsAdaptChannel::close();
        return 0;
    }

    return 1;
}

void WindowsAdaptChannel::close()
{
    flush();

    WindowsLiveChannel::close();
}



//////////////////////////////////////////////////////////////////////
// VocStateMachine
//////////////////////////////////////////////////////////////////////
int fileNameCmp( const char *f1, const char *f2 ) {
    // fname is a null terminated string
    assert( f1 && *f1 );
    assert( f2 && *f2 );

    for( const char *s1=f1; *s1; ++s1 );
    for( const char *s2=f2; *s2; ++s2 );

    while( s1>f1 && s2>f2 ) {
        --s1;    --s2;
        if( tolower( *s1 ) != tolower( *s2 ) )
            return 1;
    }

    if( s1>f1 )
        return !( *(s1-1) == ':' || *(s1-1) == '\\' || *(s1-1) == '/' );

    if( s2>f1 )
        return !( *(s2-1) == ':' || *(s2-1) == '\\' || *(s2-1) == '/' );

    return 0;
}

SD_VOC VocStateMachine::open( const char *name )
{
    // always load the user then the voc, close the voc then the user
    assert( name );
    state = NO_CURRENT_STATE;

    return setVoc( SDVoc_Open( name, "r+" ) );
}

// save Vocabulary
```

```
void VocStateMachine::save( const char *name )
{
    assert( voc );

    if( name )
    {
        char buf[ 128 ];

        getVocname( buf, sizeof( buf ) );

        if( fileNameCmp( buf, name ) )
        {
            SDVoc_SaveAs( voc, name );

            return;
        }
    }

    SDVoc_Save( voc );
}

// Close all vocabularies
void VocStateMachine::close( const char* name )
{
    SD_VOC v=0;

    if( name )
    {
        SD_VOC_ITERATOR iter;

        SDVoc_Iterate( &iter );

        while( (v=SDVoc_Next( &iter )) != 0 )
        {
            char buf[ 128 ];
            size_t len = SDVoc_GetFileName( v, buf, 128 );
            assert( len <= 128 );

            if( !fileNameCmp( name, buf ) )
                break;
        }
    } // if(name)
    else
        v = voc;

    if( v != 0 )
    {
        for( VocStateMachine *vsm=vsmDeq.first(); vsm; vsm=vsm->next() )
        {
            if( vsm->voc == v )
            {
                vsm->voc = 0;
                vsm->state = NO_CURRENT_STATE;
            }
        }

        SDVoc_Close( v );
    }
}

// Get the right Vocabulary and set it for the recognizer
SD_VOC VocStateMachine::setVoc( const char* vname )
{
    SD_VOC_ITERATOR iter;

    SD_VOC v = 0;
    SDVoc_Iterate( &iter );

        // Loop through all known Vocs and compare their name with vname
    while( (v=SDVoc_Next( &iter )) != 0 )
    {
        char buf[ 128 ];
        size_t len = SDVoc_GetFileName( v, buf, 128 );
        assert( len <= 128 );

        // if vname is found, break out of loop
        if( !fileNameCmp( vname, buf ) )
            break;
    }

        // set vocabulary
    return setVoc( v );
}

// Get name of Vocabulary
int VocStateMachine::getVocname( char* buf, int sizebuf, SD_VOC vTest )
{
    if( vTest == 0 ) vTest = voc;

    if( vTest != 0 )
    {
        return SDVoc_GetFileName( vTest, buf, sizebuf );
    }
    if( sizebuf )
        buf[ 0 ] = '\0';

    return 0;
}

//  Get state in vocabulary
SD_STATE VocStateMachine::getState( const char* name, SD_STATE parent )
{
    char astatename[256];

    int index = -1; /* Will be 0 after first line... */
    do {
        index++;
        // First time: index<-0; Others: go past '.'
        // Read through the arg, up to the next period or the end.
        // Put text in astatename.
        sscanf( (name +index), "%[^.]", astatename );
        index += strlen( astatename );

        parent = SDState_GetHandle( voc, astatename, parent );
        if ( !parent ) break;

     } while ( name[index] == '.' );

     return parent;
 }

//void VocStateMachine::updateState( SD_WORD hWord )
//{
//    assert( state );
//
//    state = SDStMach_Update( voc, state, hWord, YES );
//}
//
//
//
//
```

```
//void VocStateMachine::updateState( const RecogResult* res )
//{
//    assert( state );
//    assert( res );
//
//    int nWords = res->wordCount();
//    for ( int j=0; j < nWords; j++ ) {
//      state = SDStMach_Update( voc, state, res->wordId(j), YES );
//    }
//}


// Is word active
bool VocStateMachine::wordIsActive( SD_WORD hWord )
{
    assert( voc );
    assert( hWord );
    assert( state != NO_CURRENT_STATE );

    SD_STATE_WORD_INFO1 info;

    SDState_GetWordInfo1( voc, state, hWord, &info );

    return info.isInState && info.isActive;
}


// Jump transition for specific SD_WORD
SD_STATE VocStateMachine::jmpTransition( SD_WORD hWord )
{
    assert( wordIsActive( hWord ) );

    SD_TRANSITION tran=SDState_GetWordTransition( voc, getState(), hWord );

    if( tran.OpCode!=JMP ) return 0;

    return( tran.args.jmp.destState );
}

// Create new state
SD_STATE VocStateMachine::newState( const char *newStateName )
{
    assert( newStateName );
    assert( voc );

    SD_STATE hState=SDState_GetHandle( voc, newStateName, 0 );

    if( hState ) return 0;

    hState = SDState_New( voc, 0 );

    if( hState ) SDState_SetName( voc, hState, newStateName );

    return hState;
}

// Add State within state
void VocStateMachine::includeState( SD_STATE stateToInclude )
{
    assert( stateToInclude );
    assert( voc );
    assert( getState() );

    SDState_AddState( voc, getState(), stateToInclude, (uint)(-1) );
}
```

```
// Delete state
void VocStateMachine::deleteState( SD_STATE hState )
{
    assert( voc );
    assert( hState );

    int flag = ( hState==getState() );

    SDState_Delete( voc, hState);

    if( flag ) setState( (SD_STATE) 0 );
}


// Get first SD_WORD handle of state
SD_WORD VocStateMachine::firstWordId( SD_STATE st )
{
    if( st == (SD_STATE)(-1) ) st = getState();

    SD_WORD hWord = 0;

    stateOfIterator = st;

    // usingWordIterator = -1 none, 0 state0, state

    if( st == 0 )
    {
        SDWord_Iterate( voc, &stateOWordIterator );

        hWord = SDWord_Next( &stateOWordIterator );
    }
    else if( st > 0 )
    {
        SDState_IterateWords( voc, st, &stateWordIterator );

        hWord = SDState_NextWord( &stateWordIterator );
    }

    if( hWord == 0 )
        stateOfIterator = -1;

    return hWord;
}

// Get next word ID
SD_WORD VocStateMachine::nextWordId()
{
    SD_WORD hWord;

    if( stateOfIterator == 0 )
        hWord = SDWord_Next( &stateOWordIterator );

    else if( stateOfIterator != (unsigned )-1 )
        hWord = SDState_NextWord( &stateWordIterator );

    else
        hWord = 0;

    if( hWord == 0 )
        stateOfIterator = -1;

    return hWord;
}

// Get nth word ID in state
SD_WORD VocStateMachine::nthWordId( long index )
```

```
{
    assert( voc );

    assert( state != NO_CURRENT_STATE );

    SD_WORD hWord;

        // If there are no states in the Vocabulary
    if( state == 0 )
        SDWord_List( voc, index, &hWord, 1 );

    else
        SDState_ListWords( voc, state, SD_ORDER_ALPHA, index, &hWord, 1 );

    return hWord;
}

// List words (SD_WORD) in state
void VocStateMachine::listState( long numToList, FILE* file )
{
    assert( voc );

#define MAXWORD 256

    char wordbuf[ MAXWORD ];

    SD_STATE hState = getState();

    SD_WORD hWord;

    if( file == 0 )
        file = stdout;

        // There is no state in Vocabulary
    if( hState == 0 )
    {
        SD_WORD_ITERATOR it;

        SDWord_Iterate( voc, &it );

        if( numToList ) {
            while( numToList-- && (hWord=SDWord_Next( &it )) != 0 )
            {
                SDWord_GetName( voc, hWord, wordbuf, MAXWORD );
                SD_WORD_INFO wordInfo;
                SDWord_GetInfo( voc, hWord, &wordInfo );
                if( wordInfo.hasModel )
                    fprintf( file, "%s  %lu\n", wordbuf, (long unsigned )hWord )
=> ;
                else
                    fprintf( file, "NO MODELS for: %s   %lu\n", wordbuf, (long u
=> nsigned )hWord );
            }
        }
        else {
            unsigned i = 0;
            while( (hWord=SDWord_Next( &it )) != 0 )
            {
                if( i > 15000 )
                {
                    if( i == 16800 )
                        fprintf( file, "16800 multiple: %d\n", i );
                    else if( i == 16600 )
                        fprintf( file, "16600 multiple: %d\n", i );
                    else if( i == 16400 )
                        fprintf( file, "16400 multiple: %d\n", i );
                else if( i == 16200 )
                    fprintf( file, "16200 multiple: %d\n", i );
                else if( i == 16000 )
                    fprintf( file, "16000 multiple: %d\n", i );
                else if( i == 15800 )
                    fprintf( file, "15800 multiple: %d\n", i );
                else if( i == 15600 )
                    fprintf( file, "15600 multiple: %d\n", i );
                else if( i == 15400 )
                    fprintf( file, "15400 multiple: %d\n", i );
                else if( i == 15200 )
                    fprintf( file, "15200 multiple: %d\n", i );
                else if( i == 15000 )
                    fprintf( file, "15000 multiple: %d\n", i );
            }

            // Get word and check for model
            SDWord_GetName( voc, hWord, wordbuf, MAXWORD );
            SD_WORD_INFO wordInfo;
            SDWord_GetInfo( voc, hWord, &wordInfo );
            if( wordInfo.hasModel )
                fprintf( file, "%s\n", wordbuf );
            else
                fprintf( file, "NO MODELS for: %s\n", wordbuf );
        }
    }

  }

    // there are states in vocabularies
  else
  {
    SD_STATE_WORD_ITERATOR it;

    SDState_IterateWords( voc, hState, &it );

    if( numToList ) {
        while( numToList-- && (hWord=SDState_NextWord( &it )) != 0 )
        {
                    SDWord_GetName( voc, hWord, wordbuf, MAXWORD );
                    SD_WORD_INFO wordInfo;
                    SDWord_GetInfo( voc, hWord, &wordInfo );
                    if( wordInfo.hasModel )
                        fprintf( file, "* %s\n", wordbuf );
                    else
                    fprintf( file, "  %s\n", wordbuf );
        } // while
    } // if
    else {
        while( (hWord=SDState_NextWord( &it )) != 0 )
        {
                    SDWord_GetName( voc, hWord, wordbuf, MAXWORD );
                    SD_WORD_INFO wordInfo;
                    SDWord_GetInfo( voc, hWord, &wordInfo );
                    if( wordInfo.hasModel )
                        fprintf( file, "* %s\n", wordbuf );
                    else
                    fprintf( file, "  %s\n", wordbuf );
        } //  while
    } // else
  } // else
}

// Count number of words in vocabulary
long VocStateMachine::wordCount()
{
```

```
    assert( voc );

    assert( state != NO_CURRENT_STATE );

    if( state == 0 )
    {
        SD_VOC_INFO vInfo;

        SDVoc_GetInfo( voc, &vInfo );

        return vInfo.nWords;
    }

    SD_STATE_INFO1 info;

    SDState_GetInfo1( voc, state, &info );

    return info.nWords;
}

// Check whether word has a model
bool VocStateMachine::wordHasModel( SD_WORD w )
{
    SD_WORD_INFO wordInfo;
    if( !w ) return 0;
    SDWord_GetInfo( voc, w, &wordInfo );
    return( wordInfo.hasModel != 0 );
}

// Add word to vocabulary
SD_WORD VocStateMachine::addWord( SD_WORD w )
{
    assert( w != 0 );

    assert( voc );

    SD_STATE st = getState();

    assert( st );

    SD_STATE_WORD_INFO1 wInfo1;

    SDState_GetWordInfo1( voc, st, w, &wInfo1 );

        // Do not add word if it is already in state
    if( wInfo1.isInState )
    {
        return w;
    }

    SDState_AddWord( voc, st, w );

    return w;
}

// Add a word to the state.
SD_WORD VocStateMachine::addWord( const char* s )
{
    assert( voc );
    assert( s );

    SD_WORD w = SDWord_GetHandle( voc, s );

    if( w==0 )
    {
        w = SDWord_New( voc, s );
    }

    return addWord( w );
}

// phrase building
SD_WORD VocStateMachine::buildWord( const char* spelling, const char* phrase )
{
        // string of words not seperated by spaces
        assert( spelling );
    // phrase = words seperated by spaces null terminated
        assert( phrase );
    assert( isalnum( *phrase ) );

    char *phrCpy = new char[ strlen( phrase ) +1 ];

    char *pc = phrCpy;

    char *p = (char* )phrase;

    int nWords = 1;

    while(  (*pc = *p) != 0 )
    {
        if( *p == ' ' )
        {
            *pc = '\0';

            while( *++p == ' ' );

            ++nWords;
        }

        else if( *p == '\r' || *p == '\n' )
        {
            *pc = '\0';
            break;
        }
        else
            ++p;

        ++pc;
    }

        // build word ID for phrase, and return SD_WORD
    SD_WORD w = SDWord_BuildFrom( voc, spelling, 0, 0, nWords, phrCpy, 0, 0 );

    delete phrCpy;

    if( w )
        return addWord( w );

    return w;
}

// Delete word from vocabulary,  being given SD_WORD
bool VocStateMachine::deleteWord( SD_WORD w )
{
    SD_WORD_INFO wInfo;
    SDWord_GetInfo( voc, w, &wInfo );

    if( wInfo.stateRefCount >= 1 )
        wInfo.stateRefCount -= removeWordfromState( w );

    if( wInfo.stateRefCount == 0 )
```

DRAGCPP.CPP  7-22-95  11:35a

```
    {
        SDWord_Delete( voc, w );

        return 1;
    }
    return 0;
}

// Delete word from vocabulary,  being given name of word to delete
bool VocStateMachine::deleteWord( const char* s )
{
        //Get SD_WORD for word name
    SD_WORD w = wordId( s );
        // Delete SD_WORD from vocabulary
    return( w ? deleteWord( w ) : 0 );
}

// Delete word from state
bool VocStateMachine::removeWordfromState( SD_WORD w )
{
    SD_STATE_WORD_INFO1 wInfo;
    SDState_GetWordInfo1( voc, getState(), w, &wInfo );

    if( wInfo.isInState )
    {
        SDState_DeleteWord( voc, getState(), w );
        return 1;
    }
    return 0;
}

// Delete word from state
bool VocStateMachine::removeWordfromState( const char* s )
{
    SD_WORD w = wordId( s );

    return( w ? removeWordfromState( w ) : 0 );
}



/////////////////////////////////////////////////////////////////////
// UserPhonetics
/////////////////////////////////////////////////////////////////////

// Open USER file
SD_USER UserPhonetics::open( const char *name )
{
    // always load the user then the voc, close the voc then the user
    assert( name );

    return setUser( SDUser_Open( name ) );
}

// Set user
SD_USER UserPhonetics::setUser( SD_USER u )
{
    if( activeUser != u )
    {
        SDUser_SetCurrent( activeUser = u );
    }
    user = u;

    return user;
}
```

```
// Set user
SD_USER UserPhonetics::setUser( const char* name )
{
    SD_USER_ITERATOR iter;
    SD_USER u = 0;

    assert( name );

    SDUser_Iterate( &iter );
    while( (u=SDUser_Next( &iter )) != 0 )
    {
        char buf[ 128 ];
        size_t len = SDUser_GetFileName( u, buf, 128 );
        assert( len <= 128 );

        if( !fileNameCmp( name, buf ) )
            break;
    }

    return setUser( u );
}

// Get name of loaded user
int UserPhonetics::getUsername( char* buf, int sizebuf, SD_USER uTest )
{
    if( uTest == 0 ) uTest = user;

    if( uTest != 0 )
    {
        return SDUser_GetFileName( uTest, buf, sizebuf );
    }
    if( sizebuf )
        buf[ 0 ] = '\0';

    return 0;
}

// Save user file
void UserPhonetics::save( const char *name )
{
    assert( user );

    if( name )
    {
        char buf[ 128 ];

        getUsername( buf, sizeof( buf ) );

        if( *buf && fileNameCmp( buf, name ) )
        {
            SDUser_SaveAs( user, name );

            return;
        }
    }

    SDUser_Save( user );
}

// Close User file
void UserPhonetics::close( const char* name )
{
    SD_USER u=0;

    if( name )
```

```
    {
        SD_USER_ITERATOR iter;

        SDUser_Iterate( &iter );

        while( (u=SDUser_Next( &iter )) != 0 )
        {
            char buf[ 128 ];
            size_t len = SDUser_GetFileName( u, buf, 128 );
            assert( len <= 128 );

            if( !fileNameCmp( name, buf ) )
                break;
        }
    }
    else
        u = user;

    if( u != 0 )
    {
        for( UserPhonetics *up=upDeq.first(); up; up=up->next() )
        {
            if( up->user == u ) up->user = 0;
        }

        if( activeUser == u ) activeUser = 0;

        SDUser_Close( u );
    }
}


////////////////////////////////////////////////////////////////////////////
// RecogResult
////////////////////////////////////////////////////////////////////////////

int RecogResult::getName( char* textbuf, int sizebuf ) const
{
    if( !this ) {
        if( textbuf && sizebuf ) *textbuf='\0';
        return 0;
    }

    char *bufpos = textbuf;

    for ( int j=0; j < nWords; j++ ) {
        bufpos +=
        SDWord_GetName( wordSpec[j].hVoc,
                        wordSpec[j].hWord,
                        bufpos,
                        textbuf + sizebuf - bufpos );

        if( bufpos - textbuf > sizebuf )
            bufpos = textbuf + sizebuf;

        if( j+1 < nWords )
            *( bufpos - 1 )= ' '; // Put space between words.
    }
    return( bufpos - textbuf );
}


// Jump Transition based on Word
SD_STATE RecogResult::jmpTransition( int i ) const
{
    assert( i<wordCount() );
    SD_TRANSITION tran=SDState_GetWordTransition( vocID(i), stateID(i), wordId(i
  => )˜);

    if( tran.OpCode!=JMP ) return 0;

    return( tran.args.jmp.destState );
}

////////////////////////////////////////////////////////////////////////
// Recognizer
////////////////////////////////////////////////////////////////////////

// Constructor for Recognizer class
// You need a channel to construct this class
Recognizer::Recognizer( UttChannel* recogCh )
: userPhonetics(), vsm() {

    assert( recogCh );

    channel = recogCh;

    setPars();
};

Recognizer::˜Recognizer()
{
    channel = 0;
    setUser( (SD_USER )0 );
    setVoc( (SD_VOC )0 );
}

// Open a vocabulary
SD_VOC Recognizer::setVoc( const char *name )
{
    assert( name );

    SD_VOC v = vsm.setVoc( name );

    if( v == 0 )
    {
        v = vsm.open( name );

        if( userPhonetics.user != 0 && v != 0 )
            vsm.loadPhonetics();
    }

    return v;
}

// Set the user models
SD_USER Recognizer::setUser( const char* vname )
{
    assert( vname );

    SD_USER u = userPhonetics.setUser( vname );

    if( u == 0 )
    {
        u = userPhonetics.open( vname );

        if( u != 0 && vsm.voc != 0 )
            vsm.loadPhonetics();
    }
```

```
    return u;
}

// Discrete Recognition call
// Default passThru is 0

// Usually we pass nothing extra. The recognition is done
// from the set state. It is possible to pass in an extra list
// of words that do not belong to the state your recognizing
// from, but that you want the recognizer to consider.
SD_REJCODE Recognizer::recog( AC< SD_WORD > *passThru )
{
    assert( channel );
    assert( vsm.voc );
    assert( vsm.getState() != NO_CURRENT_STATE );

    userPhonetics.active();

    SD_UTT u;

    if( (u=channel->startUtt()) != 0 )
    {
        if( passThru )
        {
            recogPars.pPassThru = &( (*passThru)[0] );
            recogPars.nPassThru = passThru->count();
        }
        else
        {
            recogPars.pPassThru = 0;
            recogPars.nPassThru = 0;
        }

        channel->choices->isCont = 0;
        channel->choices->recogStatus.nChoices = 0;
        channel->choices->recogUser = getUser();

        // Recognition function called from SDAPI level.
        // This function is not provided with VoiceTools
        int rtn = SDState_Recogx(
                                        channel->choices->recogVoc = getVoc(),
  =>            // Voc
                                        channel->choices->recogState = getState(
  => ),         // State
                                        u,
  =>                                              // User
                                        &( channel->choices->recogResults[ 0 ] )
  => ,          // Choice list
                                        sizeof( channel->choices->recogResults )
  => ,          // sizeof list
                                        &recogPars,
  =>            // Recognition parameters
                                        &(channel->choices->recogStatus) );

  =>                    // Recognition Status structure

        channel->endUtt();

        return( !rtn );
    }

    return 1;
}
```

```
// Adapt models for discrete recognition
SD_REJCODE Recognizer::adapt()
{
    assert( channel );
    assert( vsm.voc );

    userPhonetics.active();

    const char *uttPrompt = channel->prompt();

    if( uttPrompt && *uttPrompt )
    {
        SD_WORD w = wordId( uttPrompt );

        if( w == 0 )
            return 1;

        SD_UTT u;
        SD_REJCODE rejCode;
        if( (u=channel->startUtt()) != 0 )
        {
            SDWord_Adapt( vsm.voc, 1, uttPrompt,
                          1, &u,
                          &trainPars, &( channel->choices->trainStatus ),
                          &rejCode );

            channel->endUtt();
        }

        SD_WORD hWord;

        if( adaptLm ) {
            if( (hWord=SDWord_GetHandle( vsm.voc, uttPrompt )) != 0 )
                SDWord_AdaptLm( vsm.voc, hWord, &adaptLmPars );
            else {
                // FUTURE: Optionally add word to vocabulary

#ifdef DEBUG
                printf( "\"%s\" not in vocabulary.\n", uttPrompt );
#endif

                return 1;
            }
        }
        return rejCode;
    }
    return 1;
}

// collect utterances, given a prompt
SD_REJCODE Recognizer::collect()
{
    assert( channel );
    assert( channel->uttPrompt );
    assert( channel->uttPrompt[0] );
    assert( channel->utt() == 0 );

    userPhonetics.active();

    SD_UTT u;

    if( (u=channel->startUtt()) != 0 )
    {
        SD_UTT_INFO uttInfo;

        channel->choices->isCont = 0;
```

```
        channel->choices->recogUser = getUser();
        channel->choices->recogVoc = getVoc();
        channel->choices->recogState = getState();

        SDUtt_Collect( u, &uttInfo );

        channel->endUtt();

        return 0;
    }
    return 1;
}

// check whether word is trained
bool Recognizer::wordIsTrained( SD_WORD w )
{
    SD_WORD_INFO wordInfo;
    if( !w ) return 0;
    SDWord_GetInfo( vsm.voc, w, &wordInfo );
    return( wordInfo.isTrained != 0 );
}

// check whether word is trained
bool Recognizer::wordIsTrained( const char *s )
{
    SD_WORD_INFO wordInfo;
    SD_WORD w = SDWord_GetHandle( vsm.voc, s );
    if( !w ) return 0;
    SDWord_GetInfo( vsm.voc, w, &wordInfo );
    return( wordInfo.isTrained != 0 );
}


/////////////////////////////////////////////////////////////////////////
// ContinuousRecognition
/////////////////////////////////////////////////////////////////////////

SD_REJCODE Recognizer::contRecog( int maxWords, int minWords )
{
    assert( channel );

    SD_VOC v = getVoc();
    assert( v );

    SD_STATE st = getState();
    assert( st != NO_CURRENT_STATE );

    userPhonetics.active();

    channel->choices->recogStatus.nChoices = 0;
    channel->choices->recogVoc = v;
    channel->choices->recogState = st;
    channel->choices->isCont = 1;
    channel->choices->recogUser = getUser();

    SD_UTT u;

    if( (u=channel->startUtt()) != 0 )
    {
        if( maxWords == 0 && minWords == 0 )
        {
            SD_STATE_MACHINE stMach = SDStMach_New( v, st );
                // driver function, not surfaced in SDAPI
            SDState_ContRecog( v, stMach, u,
                               &( channel->choices->recogResults[ 0 ] ),
                                          sizeof( channel->choices->recogResults ),
                                          (RECOG_PARAMETERS* )&recogPars,
                                          (RECOG_STATUS* )&(channel->choices->recogStatus)
=> );

            SDStMach_Delete( stMach );
        }
        else
                // driver function surfaced in SDAPI
            SDState_ContRecog1( v, st, u, 0, minWords, maxWords,
                                &( channel->choices->recogResults[ 0 ] ),
                                sizeof( channel->choices->recogResults ),
                                (RECOG_STATUS* )&( channel->choices->recogStatus
=>  ) );

        channel->endUtt();

        return lresultCount();
    }

    return 1;
}

// Adapt models
SD_REJCODE Recognizer::contAdapt()
{
    assert( channel );
    assert( vsm.voc );

    userPhonetics.active();

    const char *uttPrompt = channel->prompt();

    if( uttPrompt && *uttPrompt )
    {
        char str[ UTT_PROMPT_LENGTH ];

        memcpy( str, uttPrompt, UTT_PROMPT_LENGTH );

        int nWords=1;

        for( char *s=str; *s; ++s )
        {
            if( *s==' ' )
            {
                *s='\0';
                ++nWords;

                while( isspace( *(s+1) ) )          ++s;
            }
        }

        SD_UTT u;
        SD_REJCODE rejCode;
        if( (u=channel->startUtt()) != 0 )
        {
            SDWord_Adapt( vsm.voc, nWords, str, 1, &u,
                          &trainPars, &channel->choices->trainStatus,
                          &rejCode );

            channel->endUtt();
        }

        SD_WORD hWord;
```

DRAGCPP.CPP 7-22-95 11:35a

```
        if( adaptLm )
        {
            if( (hWord=SDWord_GetHandle( vsm.voc, uttPrompt )) != 0 )
                SDWord_AdaptLm( vsm.voc, hWord, &adaptLmPars );
            else {
                // FUTURE: Optionally add word to vocabulary

#ifdef DEBUG
                printf( "\"%s\" not in vocabulary.\n", uttPrompt );
#endif

                return 1;
            }
        }
        return rejCode;
    }
    return 1;
}
```

Page 17 of 17

```
/***********************************************************************
        Description

        seq.cpp
        base classes to manage singly linked list.

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995
***********************************************************************/


#include "seq.h"
#include <stddef.h>
#include <dos.h>


void SLinkBase::addNext( SLinkBase* newItem ) {
    // this is in the list, newItem is inserted at this's next
    assert( newItem != 0 );

    newItem->nextLink=nextLink;
    nextLink = newItem;
}

void SeqBase::addFirst( SLinkBase* list ) {
    //  inserts this BELOW the newItem;
    ( (list->nextLink = f) == 0
      ? (l=f=list)
      : (f=list) );
}

void SeqBase::addLast( SLinkBase* list ) {
    //  inserts this BELOW the last()
    if( l == 0 )
    {
        assert( f == 0 );
        addFirst( list );
    }
    else
    {
        list->nextLink = 0;
        l->nextLink = list;
        l = list;
    }
}

void SeqBase::addNext( SLinkBase* list, SLinkBase* newItem )
{
    if( list==l )
        addLast( newItem );

    else
        list->addNext( newItem );
}

void SeqBase::addPrev( SLinkBase* list, SLinkBase* newItem )
{
    SLinkBase *link;

    assert( list );

    if( (link=f) == list )
        addFirst( newItem );
```

```
    else
    {
        while( link->nextLink != list )
        {
            link=link->nextLink;
            assert( link );
        }

        addNext( link, newItem );
    }
}

void SeqBase::remove( SLinkBase* list )
{
    SLinkBase *link;

    assert( list );
    assert( f );

    if( (link=f) == list )
    {
        if( (f=link->nextLink) == 0 )
        {
            assert( l == list );
            l = 0;
        }
    }
    else
    {
        while( link->nextLink != list )
        {
            link=link->nextLink;
            assert( link );
        }

        if( (link->nextLink=list->nextLink) == 0 )
        {
            assert( l == list );
            l = link;
        }
    }

    #if DEBUG
    list->nextLink = 0;
    #endif
}

// void SeqBase::sort( SLinkCompare compare ) {
//}


#if DEBUG

long SeqBase::count() const {
    long i=0;
    for( SLinkBase *list=f; list; list=list->nextLink ) ++i;
    return i;
}

/*
bool SeqBase::sorted( SLinkCompare compare ) {
    SLinkBase *list=f;
    assert(list != 0);

    int i;
    while( list->nextLink != 0)
    {
        i=compare( list, list->nextLink );
        if( i > 0 )     return FALSE;
        list=list->nextLink;
    }
    return TRUE;
}
*/

void SeqBase::removeAll()
{
    SLinkBase *d, *nextL;

    for( d = f; d != 0; d = nextL )
    {
        nextL = d->nextLink;

        d->setNext( 0 );
    }

    f = l = 0;
}

bool SeqBase::validateSeq()
{
    if( f == 0 ) return( l == 0 );

    for( SLinkBase* d = f; d != 0; d = d->nextLink )
    {
        if( d == d->nextLink )
            return FALSE;

        if( d->nextLink == 0 )
            return( l == d );
    }

    return TRUE;
}

/////////////////////////////////////////////////////////////////////
//  The HashSeq class is ...
/////////////////////////////////////////////////////////////////////

/*
class HashSeqBase
{
 public:
    // constructors
    HashSeqBase( unsigned initialSize = 64 );
    // destructor
    ~HashSeqBase();


    SLinkBase* add( const SLinkBase* v );

    SLinkBase* find( SLinkBase* v );

    SLinkBase* remove( const SLinkBase* v );

    virtual int compare( const SLinkBase*, const SLinkBase* ) const = 0;

protected:
    unsigned arraySize;
    unsigned arrayUsed;
```

SEQ.CPP 7-22-95 11:36a

```
    unsigned numSLinks;

    SLinkBase** array;

    virtual unsigned keyHash( const SLinkBase* ) const = 0;

    #ifdef DEBUG
    int checkCount() const;
    #endif
};

*/


inline bool HashSeqBase::inArrayBounds( SLinkBase** pp ) const
{
#if defined(__DPMI32__)
    return( pp>=array && pp<array+arraySize );
#else
    return( FP_SEG(pp)==FP_SEG( array ) && pp>=array && pp<array+arraySize );
#endif
}


SLinkBase** HashSeqBase::first() const
{
    if( arrayUsed == 0 )        return 0;
    for( SLinkBase** pp=array, **ppe=array+arraySize; pp<ppe && !*pp; ++pp ) ;
    return pp<ppe ? pp : 0;
}


#endif
#if DEBUG || UNIT_2


SLinkBase** HashSeqBase::last() const
{
    if( arrayUsed == 0 )        return 0;

    for( SLinkBase** pp = array+arraySize-1; pp>=array && !*pp; --pp );

    assert( *pp && pp>=array );

    while( (*pp)->nextLink )
        pp = &( (*pp)->nextLink );

    return pp;
}


#endif
#if DEBUG || UNIT_3

SLinkBase** HashSeqBase::next( SLinkBase** pp ) const
{
    assert( pp != 0 );

    if( (*pp)->nextLink )
        return &( (*pp)->nextLink );

    if( !inArrayBounds( pp ) )
    {
        pp = array + ( keyHash( *pp ) & (arraySize -1) );
```

Page 2 of 5

```
        assert( pp >= array  &&  pp < array+arraySize );
    }

    for( SlinkBase **ppe=array+arraySize; ++pp<ppe && !*pp; ) ;

    return( pp < ppe ? pp : 0 );
}

#endif
#if DEBUG || UNIT_4

SLinkBase** HashSeqBase::prev( SLinkBase** pp ) const
{
    assert( pp != 0 );

    if( !inArrayBounds( pp ) )
    {
        SlinkBase **link = array + ( keyHash( *pp ) & (arraySize -1) );

        while( (*link)->nextLink != *pp )
        {
            link = &( (*link)->nextLink );
            assert( *link );
        }

        return link;
    }

    while( --pp>=array && !*pp ) ;

    if( pp < array )
        return 0;

    while( (*pp)->nextLink )
    {
        pp = &( (*pp)->nextLink );
    }

    return pp;
}

#endif
#if DEBUG || UNIT_5

SLinkBase* HashSeqBase::add( SLinkBase* v )
{
    assert( v );

    if( numSLinks > arraySize && arraySize < arrayMaxSize )
        growArray();

    unsigned hash = keyHash( v ) & (arraySize-1);

    if( array[ hash ] != 0 )
    {
        for( SLinkBase *p=array[ hash ]; p; p=p->next() )
            if( !compare( v, p ) )
                return p;
    }
    else
        ++arrayUsed;
```

```
    ++numSLinks;

    v->nextLink = array[ hash ];

    array[ hash ] = v;

    return v;
}

#endif
#if DEBUG || UNIT_6

SLinkBase* HashSeqBase::find( const SLinkBase* v )
{
    unsigned hash = keyHash( v ) & ( arraySize -1 );

    if( array[ hash ] != 0 )
    {
        for( SLinkBase *p=array[ hash ]; p && compare( v, p ); p=p->next() ) ;

        return p;
    }

    return 0;
}


#endif
#if DEBUG || UNIT_7

HashSeqBase::HashSeqBase( unsigned maxSize, unsigned initialSize  )
{
    if( initialSize <= 0 )
        arraySize = 64;

    else {
        arraySize = 1;

        for( unsigned i=initialSize; i>>=1; arraySize<<=1 );

        if( arraySize < initialSize ) arraySize<<=1;
    }

    arrayMaxSize = ( maxSize > arraySize ) ? maxSize : arraySize;


    assert( arraySize < UINT_MAX/sizeof( void* ) );

    array = new SLinkBase*[ arraySize ];

    memset( array, 0, arraySize * sizeof( SLinkBase* ) );

    arrayUsed = 0;

    numSLinks = 0;
}

#endif
#if DEBUG || UNIT_8
```

```
SLinkBase* HashSeqBase::remove( const SLinkBase* v )
{
    unsigned hash = keyHash( v ) & ( arraySize -1 );

    SLinkBase *p=array[ hash ];

    if( p != 0 )
    {
        if( v == p )
        {
            if( (array[ hash ] = p->next()) == 0 )
                --arrayUsed;

            --numSLinks;

            #if DEBUG
            p->nextLink = 0;
            #endif

            return p;
        }
        for( SLinkBase *pNext = p->next(); pNext && v != pNext; )
        {
            p = pNext;
            pNext = pNext->next();
        }

        if( pNext )
        {
            p->nextLink = pNext->nextLink;

            --numSLinks;

            #if DEBUG
            pNext->nextLink = 0;
            #endif

            return pNext;
        }
    }
    return 0;
}

#endif
#if DEBUG || UNIT_9

void HashSeqBase::growArray()
{
    if( arraySize >= arrayMaxSize )
        return;

    unsigned oldSize = arraySize;

    SLinkBase** oldArray = array;

    arraySize <<= 1;

    array = new SLinkBase*[ arraySize ];

    assert( array );

    memset( array, 0, arraySize * sizeof( SLinkBase* ) );

    arrayUsed = 0;
```

SEQ.CPP 7-22-95 11:36a

```
    numSLinks = 0;

    for( unsigned i=0; i<oldSize; ++i )
    {
        for( SLinkBase* slb=oldArray[ i ]; slb; )
        {
            SLinkBase* slbnext = slb->next();
            add( slb );
            slb = slbnext;
        }
    }

    delete [] oldArray;
}


#endif
#if DEBUG || UNIT_10

unsigned stringHash( const char* s )
{
    static unsigned randTable[] =
    {
        0x015a, 0x2ae6, 0x2d88, 0x44bb, 0xd9a4, 0x232c, 0x8df3, 0x483c,
        0x9524, 0xd7bf, 0xe9ef, 0x1c16, 0xf9ee, 0xf62c, 0xb9bb, 0xd4b1,
        0xc64b, 0x0e96, 0x8f8b, 0x3cc9, 0x4839, 0x6f32, 0x1704, 0x58ea,
        0xc4e7, 0x4fcb, 0x64c3, 0x3b04, 0xb03b, 0x4d3e, 0x95d5, 0x25c2,
        0x344e, 0x34fd, 0xbf3e, 0x0035, 0x08ed, 0xa08d, 0x2d74, 0x193c,
        0x1b18, 0xe10a, 0xf3e3, 0xbb0c, 0x0484, 0x3ea0, 0x4c69, 0x8eab,
        0xd31b, 0x7012, 0xc6d9, 0xcf77, 0x2078, 0x1e35, 0x487a, 0x4e9e,
        0x796e, 0xe774, 0xdb87, 0xc6a6, 0x1580, 0xef4c, 0xceaf, 0xfea4,
        0xde68, 0xa5cf, 0xa495, 0x3f07, 0x4793, 0x6322, 0x534f, 0x128c,
        0xf5a1, 0xd301, 0xd3e8, 0xdc49, 0xb5e7, 0xb038, 0x81e0, 0x39e5,
        0x45ef, 0xd02a, 0xcaa9, 0x375c, 0xeaf5, 0x7e4c, 0xad2a, 0xd27e,
        0x5969, 0xa7a8, 0x893d, 0x4e6f, 0xec72, 0xd65d, 0x3f33, 0x5726,
        0xad67, 0xfd1c, 0x1f4d, 0x74ef, 0xab56, 0x12aa, 0x2943, 0xcfab,
        0x227f, 0x7d63, 0x1bbf, 0xc38e, 0x73d8, 0xceb3, 0x53e0, 0xc21e,
        0xec89, 0xce9d, 0xfcba, 0x0838, 0x5d6e, 0x5736, 0x0ed2, 0xa04d,
        0x829a, 0x0028, 0x9fa4, 0x361f, 0x3ad1, 0x1a32, 0x0120, 0x3a46,
        0x0f0b, 0x7ea4, 0x3126, 0xd5af, 0x09f6, 0x16e6, 0x1910, 0x2d1a,
        0x5f73, 0x6f10, 0x9d27, 0xf499, 0xe416, 0x4dd1, 0xfc29, 0xd316,
        0x54a7, 0x4b2b, 0x7ecc, 0x95cc, 0x6da6, 0x30b3, 0x7734, 0x33cb,
        0x52c1, 0xa0b3, 0x907d, 0xa176, 0x465d, 0x9289, 0xee36, 0xf406,
        0x3115, 0x1e28, 0x9ba1, 0xd507, 0xf934, 0x1794, 0xcc76, 0x45d7,
        0xaa3c, 0x7a68, 0x69e0, 0xb32d, 0xec60, 0xa552, 0xf47c, 0x588e,
        0x4c0c, 0xa594, 0xb29f, 0xf3d7, 0x515a, 0xd283, 0xb00f, 0x48b8,
        0xe79c, 0xf909, 0x8d87, 0x7435, 0x14d7, 0x5724, 0xa035, 0x9025,
        0x0144, 0x2767, 0x613d, 0xa6b5, 0x4ecf, 0xfc76, 0x2d36, 0x75e4,
        0xc09a, 0xac8c, 0x53aa, 0x0307, 0xb279, 0x0cf7, 0xf698, 0xfe44,
        0xe075, 0x7d98, 0x39f4, 0xf619, 0xfe4b, 0x4466, 0x4323, 0xdad4,
        0x9eed, 0x38ea, 0x8881, 0xd21b, 0xebfe, 0x3fc2, 0x70da, 0x5e63,
        0xad58, 0x9621, 0xc2fa, 0x3e7b, 0x2087, 0x6d4b, 0xa50f, 0x5900,
        0x204d, 0x561b, 0x6c45, 0x27e8, 0x9c1e, 0xfc7f, 0xfc3d, 0x2ffa,
        0x5fa4, 0x32f8, 0xf689, 0x3052, 0x2a3a, 0x4e1d, 0xfa31, 0xa5e0,
        0x9673, 0xd017, 0xb32e, 0x9ee8, 0x5192, 0x6424, 0x79ef, 0xda81
    };

    for( unsigned hash=0; *s; s++ )
    {
        hash += (unsigned) *s;

        hash ^= randTable[ hash & 0xff ];
    }
```

Page 4 of 5

```
    return hash;
}


#endif

#if DEBUG
int HashSeqBase::checkCount() const {
    long cnt=0;

    for( SlinkBase** p=first(); p; p=next(p) )
        ++cnt;

    return cnt==numSlinks;
}
#endif
```

SEQ.CPP 7-22-95 11:36a

Page 5 of 5

```
/*
   slidectl.cpp : dragon horizontal slide control

   PROJECT: DragonDictate for Windows -- Custom Controls
   CREATED: February 4, 1994
   AUTHOR:  Joel Gould

   (C) Copyright Dragon Systems, Inc. 1993-1994.
   All Rights Reserved.

   ** DRAGON SYSTEMS CONFIDENTIAL **

.............................................................................

   DESCRIPTION:

   When we have the focus, we draw a flashing image over the slide
   control.  If the mouse is down, we move the flash image to correspond with
   the current mouse position.  The FlashState stored in the window extra
   data area represents the current flash state and position as follows:

        FlashState = -1      no flash drawn
        FlashState >= 0      draw flash over offset FlashState

   The current position, minimum position and maximum position are also
   stored as variables in the window extra data area.

   We also store the number of tic marks (less one) in the window extra
   data area since this is complicated to compute so we do not want to
   compute it every paint operation.

   Note that we have three different representations of horizontal location:

        position     => integer between minimum range value and maximum range
                        value inclusive which is how the dialog box sees the
                        slider position

        offset       => normalized position with the leftmost (minimum) value
                        set to zero

        pixel        => actual horizontal pixel offset from the left edge of
                        the window

   The drawing dimensions of all the components of the slider are based on
   the height of the window.  This keeps the slider itself well proportioned.
   To avoid drawing errors, all internal routines assume a minumum window
   size (10x10) even if the actual window is smaller.

.............................................................................

   TO DO LIST:

.............................................................................
*/

#include <windows.h>
#include <windowsx.h>
//#include <common.h>
#include <custcntl.h>
#include <mem.h>
#include "slidectl.h"

#define BLACK        RGB(   0,   0,   0 )
#define DARKRED      RGB( 128,   0,   0 )
#define DARKGREEN    RGB(   0, 128,   0 )
#define DARKYELLOW   RGB( 128, 128,   0 )
```

SLIDECTL.CPP 7-22-95 11:29a

```
#define DARKBLUE     RGB(   0,   0, 128 )
#define DARKMAGENTA  RGB( 128,   0, 128 )
#define DARKCYAN     RGB(   0, 128, 128 )
#define DARKGRAY     RGB( 128, 128, 128 )
#define LIGHTGRAY    RGB( 192, 192, 192 )
#define LIGHTRED     RGB( 255,   0,   0 )
#define LIGHTGREEN   RGB(   0, 255,   0 )
#define LIGHTYELLOW  RGB( 255, 255,   0 )
#define LIGHTBLUE    RGB(   0,   0, 255 )
#define LIGHTMAGENTA RGB( 255,   0, 255 )
#define LIGHTCYAN    RGB(   0, 255, 255 )
#define WHITE        RGB( 255, 255, 255 )


#define WM_SETSCROLLPOS   WM_USER+0
#define WM_GETSCROLLPOS   WM_USER+1
#define WM_SETSCROLLRANGE WM_USER+2
#define WM_GETSCROLLRANGE WM_USER+3

#define HANDLE_WM_SETSCROLLPOS(hwnd, wParam, lParam, fn) \
    ((fn)((hwnd), (int)(wParam), (BOOL)LOWORD(lParam)), 0L)

#define HANDLE_WM_SETSCROLLRANGE(hwnd, wParam, lParam, fn) \
    ((fn)((hwnd), (int)LOWORD(lParam), (int)HIWORD(lParam), (BOOL)(wParam)), 0L)

#define HANDLE_WM_GETSCROLLPOS(hwnd, wParam, lParam, fn) \
    (LRESULT)(DWORD)(int)(fn)((hwnd))

#define HANDLE_WM_GETSCROLLRANGE(hwnd, wParam, lParam, fn) \
    (LRESULT)(long)(fn)((hwnd))

// global variables

extern HINSTANCE hinstance;

// In the current implementation we do NOT use the default system colors for
// controls but rather use the standard 3D gray colors.

const COLORREF backgroundColor = WHITE;         // LIGHTGRAY;
const COLORREF highlightColor  = WHITE;
const COLORREF shadowColor     = DARKGRAY;
const COLORREF borderColor     = BLACK;
const COLORREF textColor       = BLACK;

// Timer constants

const UINT TIMER_ID   = 126;        // some random number
const UINT TIMER_RATE = 200;        // 1/5 second

// drawing dimensions

    // one half the slide width based on height
inline int slideRatio( int height ) { return height * 3 / 10; }

    // height of tic marks
inline int ticHeight( int height )  { return height * 1 / 5; }

    // height of slider pointer
inline int markHeight( int height )  { return height * 2 / 5; }

    // top row of the horizontal groove
inline int grooveTop( int height )  { return height * 1 / 3 - 1; }

    // minimum distance between tic marks
inline int minTicDist( int height ) { return height; }
```

Page 1 of 7

```
    // gap between scale and bottom
const int scaleUpAmt = 0;

    // gap between slider and bottom
const int sliderUpAmt = 2;

    // when moving mouse, this is the extra space to the left and the right
    // which we allow the mouse to be without pretending that the mouse is
    // outside the slide control
const int overhang = 10;

//////////////////////////////////////////////////////////////////////
//
// Inline functions
//

inline int DgnSlideControl::GetFlashState( HWND hWnd )
    { return GetWindowWord( hWnd, m_myOffset + 0*sizeof(int) ); }

inline void DgnSlideControl::SetFlashState( HWND hWnd, int nValue )
    { SetWindowWord( hWnd, m_myOffset + 0*sizeof(int), nValue ); }

inline int DgnSlideControl::GetScrollPos( HWND hWnd )
    { return GetWindowWord( hWnd, m_myOffset + 1*sizeof(int) ); }

void DgnSlideControl::SetScrollPos( HWND hWnd, int nValue )
    { SetWindowWord( hWnd, m_myOffset + 1*sizeof(int), nValue ); }

inline int DgnSlideControl::GetScrollMin( HWND hWnd )
    { return GetWindowWord( hWnd, m_myOffset + 2*sizeof(int) ); }

void DgnSlideControl::SetScrollMin( HWND hWnd, int nValue )
    { SetWindowWord( hWnd, m_myOffset + 2*sizeof(int), nValue ); }

inline int DgnSlideControl::GetScrollMax( HWND hWnd )
    { return GetWindowWord( hWnd, m_myOffset + 3*sizeof(int) ); }

void DgnSlideControl::SetScrollMax( HWND hWnd, int nValue )
    { SetWindowWord( hWnd, m_myOffset + 3*sizeof(int), nValue ); }

inline int DgnSlideControl::GetTicCount( HWND hWnd )
    { return GetWindowWord( hWnd, m_myOffset + 4*sizeof(int) ); }

inline void DgnSlideControl::SetTicCount( HWND hWnd, int nValue )
    { SetWindowWord( hWnd, m_myOffset + 4*sizeof(int), nValue ); }

//////////////////////////////////////////////////////////////////////
//
// Computations and Updates
//

//-------------------------------------------------------------------

void DgnSlideControl::GetMeasurements( HWND hWnd,
                                       int & totalUnits, int & halfWidth,
                                       int & fullHeight, int & rangeWidth )
{
    RECT rect;
    GetWindowRect( hWnd, &rect );

    int width = rect.right - rect.left;
    if( width < 10 )
        width = 10;

    fullHeight = rect.bottom - rect.top;
    if( fullHeight < 10 )
```

```
        fullHeight = 10;

    totalUnits = GetScrollMax( hWnd ) - GetScrollMin( hWnd ) + 1;
    if( totalUnits < 2 )
        totalUnits = 2;

    halfWidth  = slideRatio( fullHeight );
    if( halfWidth < 4 )
        halfWidth = 4;

    rangeWidth = width - 2 * halfWidth;
    if( rangeWidth < 2 ) {
        halfWidth  = ( width - 2 ) / 2;
        rangeWidth = width - 2 * halfWidth;
    }
}

//-------------------------------------------------------------------

int DgnSlideControl::ComputeOffset( HWND hWnd, int xPixel )
{
    int totalUnits;
    int halfWidth;
    int fullHeight;
    int rangeWidth;
    GetMeasurements( hWnd, totalUnits, halfWidth, fullHeight, rangeWidth );

    int nOffset = (long)xPixel * (long)(totalUnits - 1) / (long)rangeWidth;

    if( nOffset < 0 )
        return 0;
    else
    if( nOffset >= totalUnits )
        return totalUnits - 1;
    else
        return nOffset;
}

//-------------------------------------------------------------------

int DgnSlideControl::ComputePixel( HWND hWnd, int nOffset )
{
    int totalUnits;
    int halfWidth;
    int fullHeight;
    int rangeWidth;
    GetMeasurements( hWnd, totalUnits, halfWidth, fullHeight, rangeWidth );

    int nPixel = (long)nOffset * (long)rangeWidth /
    (long)(totalUnits - 1);

    return nPixel + halfWidth;
}

//-------------------------------------------------------------------

void DgnSlideControl::UpdateFlash( HWND hWnd, int nOffset )
{
    int curFlash = GetFlashState( hWnd );

    if( curFlash == nOffset )
        return;

    HDC hDC = GetDC( hWnd );
    DrawFlash( hWnd, hDC, curFlash, FALSE );
    DrawFlash( hWnd, hDC, nOffset,  TRUE );
    ReleaseDC( hWnd, hDC );

    SetFlashState( hWnd, nOffset );
}

//-----------------------------------------------------------------

void DgnSlideControl::UpdatePosition( HWND hWnd, int nPos )
{
    if( nPos == GetScrollPos( hWnd ) )
        return;

    SetFlashState( hWnd, -1 );
    OnSetScrollPos( hWnd, nPos, TRUE );

    PostMessage( GetParent( hWnd ), WM_COMMAND,
                 GetDlgCtrlID( hWnd ), MAKELONG( hWnd, BN_CLICKED ) );
}

//-----------------------------------------------------------------
//
// The tic ratio is computed based on the size of the window and the range.
// We want to have a integral number of positions per tic mark, the most tic
// marks possible assuming a minimum distance between tics.
//

void DgnSlideControl::UpdateTicCount( HWND hWnd )
{
    int totalUnits;
    int halfWidth;
    int fullHeight;
    int rangeWidth;
    GetMeasurements( hWnd, totalUnits, halfWidth, fullHeight, rangeWidth );

    // estimate the maximum number of tic marks we can fit

    int ticCount = rangeWidth / minTicDist( fullHeight );

    // if we need fewer tics, we are done

    if( ticCount >= totalUnits-1 ) {
        SetTicCount( hWnd, totalUnits-1 );
        return;
    }

    // otherwise, start decreasing the number of tic marks until we find a
    // value which is an even divisor of the total units

    for( ; ticCount > 1; ticCount-- )
        if( (totalUnits-1) % ticCount == 0 ) {
            SetTicCount( hWnd, ticCount );
            return;
        }

    // otherwise, we have only one tic mark

    SetTicCount( hWnd, 1 );
}

/////////////////////////////////////////////////////////////////
//
// Message handlers
//

//-----------------------------------------------------------------
```

SLIDECTL.CPP 7-22-95 11:29a

```
#define handleMessage( message, fn ) \
 case (message) : \
    retVal = HANDLE_ ## message ( hWnd, wParam, lParam, fn ); \
    break;

LRESULT DgnSlideControl::wndProc( HWND hWnd, UINT message,
                                  WPARAM wParam, LPARAM lParam )
{
    LRESULT retVal = 0;

    BOOL oldMessageHandled = m_bMessageHandled;
    m_bMessageHandled = TRUE;

    switch( message )
    {
        handleMessage( WM_CREATE,         OnCreate );
        handleMessage( WM_NCDESTROY,      OnNCDestroy );
        handleMessage( WM_SETFOCUS,       OnSetFocus );
        handleMessage( WM_KILLFOCUS,      OnKillFocus );
        handleMessage( WM_PAINT,          OnPaint );
        handleMessage( WM_ERASEBKGND,     OnEraseBkGnd );
        handleMessage( WM_TIMER,          OnTimer );
        handleMessage( WM_SIZE,           OnSize );
        handleMessage( WM_SETSCROLLPOS,   OnSetScrollPos );
        handleMessage( WM_SETSCROLLRANGE, OnSetScrollRange );
        handleMessage( WM_GETSCROLLPOS,   OnGetScrollPos );
        handleMessage( WM_GETSCROLLRANGE, OnGetScrollRange );
        handleMessage( WM_GETDLGCODE,     OnGetDlgCode );
        handleMessage( WM_LBUTTONDOWN,    OnLButtonDown );
        handleMessage( WM_LBUTTONDBLCLK,  OnLButtonDown );
        handleMessage( WM_LBUTTONUP,      OnLButtonUp );
        handleMessage( WM_MOUSEMOVE,      OnMouseMove );
        handleMessage( WM_KEYDOWN,        OnKeyDown );

     default:
        m_bMessageHandled = FALSE;
    }

    BOOL bTemp = m_bMessageHandled;
    m_bMessageHandled = oldMessageHandled;

    if( bTemp )
        return retVal;
    else
        return CallWindowProc( m_myDefProc, hWnd, message, wParam, lParam );
}

//-----------------------------------------------------------------

int DgnSlideControl::OnCreate( HWND hWnd, CREATESTRUCT * /*lpCreateStruct*/ )
{
    // initialize our state variables

    SetFlashState( hWnd, -1 );
    SetScrollPos( hWnd, 0 );
    SetScrollMin( hWnd, 0 );
    SetScrollMax( hWnd, 9 );

    UpdateTicCount( hWnd );

    // call the standard control initialization

    m_bMessageHandled = FALSE;
    return 0;
}
```

Page 3 of 7

```
//--------------------------------------------------------------------

void DgnSlideControl::OnNCDestroy( HWND /*hWnd*/ )
{
    // free the GDI objects the last time a control is destroyed

    m_bMessageHandled = FALSE;
}

//--------------------------------------------------------------------

void DgnSlideControl::OnSetFocus( HWND hWnd, HWND hWndOldFocus )
{
    if( hWnd == hWndOldFocus )
        return;

    SetTimer( hWnd, TIMER_ID, TIMER_RATE, 0 );
}

//--------------------------------------------------------------------

void DgnSlideControl::OnKillFocus( HWND hWnd, HWND hWndNewFocus )
{
    if( hWnd == hWndNewFocus )
        return;

    UpdateFlash( hWnd, -1 );
    KillTimer( hWnd, TIMER_ID );
}

//--------------------------------------------------------------------

void DgnSlideControl::OnTimer( HWND hWnd, UINT /*id*/ )
{
    if( hWnd != GetFocus( ) || hWnd == GetCapture( ) )
        return;

    if( GetFlashState( hWnd ) < 0 )
        UpdateFlash( hWnd, GetScrollPos( hWnd ) - GetScrollMin( hWnd ) );
    else
        UpdateFlash( hWnd, -1 );
}

//--------------------------------------------------------------------

void DgnSlideControl::OnSize( HWND hWnd, UINT /*state*/,
                              int /*cx*/, int /*cy*/ )
{
    UpdateTicCount( hWnd );
}

//--------------------------------------------------------------------

void DgnSlideControl::OnSetScrollPos( HWND hWnd, int nValue, BOOL fRedraw )
{
    int curPos = GetScrollPos( hWnd );
    int curMin = GetScrollMin( hWnd );
    int curMax = GetScrollMax( hWnd );

    if( nValue < curMin )
        nValue = curMin;
    if( nValue > curMax )
        nValue = curMax;

    SetScrollPos( hWnd, nValue );
    if( fRedraw && nValue != curPos ) {
```

SLIDECTL.CPP 7-22-95 11:29a

```
        InvalidateRect( hWnd, 0, TRUE );
        UpdateWindow( hWnd );
    }
}

//--------------------------------------------------------------------

void DgnSlideControl::OnSetScrollRange( HWND hWnd, int nMin, int nMax,
                                        BOOL fRedraw )
{
    int curPos = GetScrollPos( hWnd );
    int curMin = GetScrollMin( hWnd );
    int curMax = GetScrollMax( hWnd );

    if( nMax < nMin + 1 )
        nMax = nMin + 1;

    SetScrollMin( hWnd, nMin );
    SetScrollMax( hWnd, nMax );

    if( curPos < nMin )
        SetScrollPos( hWnd, nMin );
    if( curPos > nMax )
        SetScrollPos( hWnd, nMax );

    UpdateTicCount( hWnd );

    if( fRedraw && ( nMin != curMin || nMax != curMax ||
                     GetScrollPos( hWnd ) != curPos ) ) {
                        InvalidateRect( hWnd, 0, TRUE );
                        UpdateWindow( hWnd );
    }
}

//--------------------------------------------------------------------

int DgnSlideControl::OnGetScrollPos( HWND hWnd )
{
    return GetScrollPos( hWnd );
}

//--------------------------------------------------------------------

long DgnSlideControl::OnGetScrollRange( HWND hWnd )
{
    return MAKELONG( GetScrollMin( hWnd ),
                     GetScrollMax( hWnd ) );
}

//--------------------------------------------------------------------

UINT DgnSlideControl::OnGetDlgCode( HWND /*hWnd*/, MSG FAR* /*lpmsg*/ )
{
    return DLGC_WANTARROWS;
}

//--------------------------------------------------------------------
//
// A single button down always sets the focus.  Over the slide we also set
// the capture for moving the slide.  A double click sets the capture
// whether or not we are over the slide.
//

void DgnSlideControl::OnLButtonDown( HWND hWnd, BOOL fDoubleClick,
                                     int x, int /*y*/, UINT /*keyFlags*/ )
{
```

Page 4 of 7

```
    if( hWnd == GetCapture( ) ) {
        UpdateFlash( hWnd, ComputeOffset( hWnd, x ) );
        return;
    }

    SetFocus( hWnd );

    // we test to see if the button is over the slide; we do not test to see
    // if the buttons position matches the slides position since that may
    // fail on small sliders (where separate positions are very close).

    int totalUnits;
    int halfWidth;
    int fullHeight;
    int rangeWidth;
    GetMeasurements( hWnd, totalUnits, halfWidth, fullHeight, rangeWidth );

    int xCenter =
    ComputePixel( hWnd, GetScrollPos( hWnd ) - GetScrollMin( hWnd ) );

    if( fDoubleClick ||
        ( x >= xCenter - halfWidth - 1 && x < xCenter + halfWidth + 1 ) ) {
            SetCapture( hWnd );
            UpdateFlash( hWnd, ComputeOffset( hWnd, x ) );
    }
}
//-----------------------------------------------------------------------

void DgnSlideControl::OnLButtonUp( HWND hWnd, int x, int y, UINT /*keyFlags*/ )
{
    if( hWnd != GetCapture( ) )
        return;

    ReleaseCapture( );

    RECT rect;
    GetWindowRect( hWnd, &rect );
    int width  = rect.right - rect.left;
    int height = rect.bottom - rect.top;

    if( x < 0-overhang || x >= width+overhang || y < 0 || y >= height )
        UpdateFlash( hWnd, GetScrollPos( hWnd ) - GetScrollMin( hWnd ) );

    else {
        int oldPos = GetScrollPos( hWnd );
        int newPos = ComputeOffset( hWnd, x ) + GetScrollMin( hWnd );

        if( oldPos == newPos )
            UpdateFlash( hWnd, -1 );

        else {
            UpdatePosition( hWnd, newPos );
        }
    }
}
//-----------------------------------------------------------------------

void DgnSlideControl::OnMouseMove( HWND hWnd, int x, int y, UINT /*keyFlags*/ )
{
    if( hWnd != GetCapture( ) )
        return;

    RECT rect;
    GetWindowRect( hWnd, &rect );
```

```
    int width  = rect.right - rect.left;
    int height = rect.bottom - rect.top;

    if( x < 0-overhang || x >= width+overhang || y < 0 || y >= height )
        UpdateFlash( hWnd, GetScrollPos( hWnd ) - GetScrollMin( hWnd ) );
    else
        UpdateFlash( hWnd, ComputeOffset( hWnd, x ) );
}
//-----------------------------------------------------------------------

void DgnSlideControl::OnKeyDown( HWND hWnd, UINT vk, BOOL /*fDown*/,
                                 int /*cRepeat*/, UINT /*flags*/ )
{
    int curPos = GetScrollPos( hWnd );
    int curMin = GetScrollMin( hWnd );
    int curMax = GetScrollMax( hWnd );
    int pageSize = ( curMax - curMin ) / GetTicCount( hWnd );

    switch( vk ) {

     case VK_LEFT:
     case VK_UP:
        if( curPos > curMin )
            UpdatePosition( hWnd, curPos - 1 );
        return;

     case VK_RIGHT:
     case VK_DOWN:
        if( curPos < curMax )
            UpdatePosition( hWnd, curPos + 1 );
        return;

     case VK_HOME:
        UpdatePosition( hWnd, curMin );
        return;

     case VK_END:
        UpdatePosition( hWnd, curMax );
        return;

     case VK_PRIOR:     // page up
        if( curPos > curMin )
            UpdatePosition( hWnd, curPos - pageSize );
        return;

     case VK_NEXT:      // page down
        if( curPos < curMax )
            UpdatePosition( hWnd, curPos + pageSize );
        return;

    } // switch
}
//-----------------------------------------------------------------------

void DgnSlideControl::DrawFlash( HWND hWnd, HDC hDC, int nOffset,
                                 BOOL /*bState*/ )
{
    if( nOffset < 0 )
        return;

    int totalUnits;
    int halfWidth;
    int fullHeight;
    int rangeWidth;
    GetMeasurements( hWnd, totalUnits, halfWidth, fullHeight, rangeWidth );

    int xCenter = ComputePixel( hWnd, nOffset );

    RECT rect;
    rect.left   = xCenter - halfWidth;
    rect.right  = xCenter + halfWidth;
    rect.top    = 0;
    rect.bottom = fullHeight - sliderUpAmt;
    DrawFocusRect( hDC, &rect );
}

//-----------------------------------------------------------------------

void DgnSlideControl::OnPaint( HWND hWnd )
{
    PAINTSTRUCT paintStruct;
    HDC hDC = BeginPaint( hWnd, &paintStruct );

    // create GDI objects
    HPEN hBorderPen = CreatePen( PS_SOLID, 1, borderColor );
    HPEN hHighlightPen = CreatePen( PS_SOLID, 1, highlightColor );
    HPEN hShadowPen = CreatePen( PS_SOLID, 1, shadowColor );

    // get the drawing dimensions

    int totalUnits;
    int halfWidth;
    int fullHeight;
    int rangeWidth;
    GetMeasurements( hWnd, totalUnits, halfWidth, fullHeight, rangeWidth );

    int xCenter =
    ComputePixel( hWnd, GetScrollPos( hWnd ) - GetScrollMin( hWnd ) );

    // draw the shadow for the slider

    HPEN oldPen = (HPEN) SelectObject( hDC, hShadowPen );

    MoveTo( hDC, xCenter + halfWidth, 2 );
    LineTo( hDC, xCenter + halfWidth, fullHeight - sliderUpAmt - halfWidth );
    LineTo( hDC, xCenter,             fullHeight - sliderUpAmt );

    // draw the bottom scale with tic marks

    SelectObject( hDC, hBorderPen );

    MoveTo( hDC, halfWidth - 1,              fullHeight - scaleUpAmt - 1 );
    LineTo( hDC, halfWidth + rangeWidth - 1, fullHeight - scaleUpAmt - 1 );

    int ticCount = GetTicCount( hWnd );
    for( int i = 0; i <= ticCount; i++ ) {
        MoveTo( hDC, halfWidth + i * rangeWidth / ticCount - 1,
                fullHeight - scaleUpAmt - 1 );
        LineTo( hDC, halfWidth + i * rangeWidth / ticCount - 1,
                fullHeight - scaleUpAmt - 1 - ticHeight(fullHeight) );
    }

    // draw the groove

    int width = 2 * halfWidth + rangeWidth;
    int top   = grooveTop(fullHeight);

    SelectObject( hDC, hBorderPen );

    MoveTo( hDC, 0,         top );
```

```
    LineTo( hDC, width - 1, top );
    LineTo( hDC, width - 1, top + 3 );
    LineTo( hDC, 0,         top + 3 );
    LineTo( hDC, 0,         top );

    SelectObject( hDC, hHighlightPen );

    MoveTo( hDC, 2,         top + 2 );
    LineTo( hDC, width - 2, top + 2 );
    LineTo( hDC, width - 2, top + 1 );

    SelectObject( hDC, hShadowPen );

    MoveTo( hDC, 1,         top + 2 );
    LineTo( hDC, 1,         top + 1 );
    LineTo( hDC, width - 1, top + 1 );

    // put back the slider shadow over the white groove point

    SelectObject( hDC, hShadowPen );

    MoveTo( hDC, xCenter + halfWidth, top + 2 );
    LineTo( hDC, xCenter + halfWidth, top + 3 );

    // draw the slider

    SelectObject( hDC, hBorderPen );

    MoveTo( hDC, xCenter - halfWidth + 1, 0 );
    LineTo( hDC, xCenter + halfWidth - 2, 0 );
    LineTo( hDC, xCenter + halfWidth - 1, 1 );
    LineTo( hDC, xCenter + halfWidth - 1, fullHeight - sliderUpAmt - halfWidth )
=> ;
    LineTo( hDC, xCenter,                 fullHeight - sliderUpAmt - 1 );
    LineTo( hDC, xCenter - 1,             fullHeight - sliderUpAmt - 1 );
    LineTo( hDC, xCenter - halfWidth,     fullHeight - sliderUpAmt - halfWidth )
=> ;
    LineTo( hDC, xCenter - halfWidth,     1 );
    LineTo( hDC, xCenter - halfWidth + 1, 0 );

    SelectObject( hDC, hHighlightPen );

    MoveTo( hDC, xCenter - halfWidth + 1, fullHeight - sliderUpAmt - halfWidth )
=> ;
    LineTo( hDC, xCenter - halfWidth + 1, 1 );
    LineTo( hDC, xCenter + halfWidth - 1, 1 );

    MoveTo( hDC, xCenter - halfWidth + 2, fullHeight - sliderUpAmt - halfWidth +
=>  1 );
    LineTo( hDC, xCenter - halfWidth + 2, 2 );
    LineTo( hDC, xCenter + halfWidth - 2, 2 );

    SelectObject( hDC, hShadowPen );

    MoveTo( hDC, xCenter + halfWidth - 2, 2 );
    LineTo( hDC, xCenter + halfWidth - 2, fullHeight - sliderUpAmt - halfWidth )
=> ;
    LineTo( hDC, xCenter,                 fullHeight - sliderUpAmt - 2 );

    MoveTo( hDC, xCenter + halfWidth - 3, 3 );
    LineTo( hDC, xCenter + halfWidth - 3, fullHeight - sliderUpAmt - halfWidth )
=> ;
    LineTo( hDC, xCenter,                 fullHeight - sliderUpAmt - 3 );

    // draw the pointer part of the slider
    SelectObject( hDC, hBorderPen );

    MoveTo( hDC, xCenter - 1, fullHeight - sliderUpAmt - markHeight( fullHeight
  => ) );
    LineTo( hDC, xCenter - 1, fullHeight - sliderUpAmt - 1 );

    // erase groove in center of slider

    RECT rect;
    rect.left   = xCenter - halfWidth + 3;
    rect.right  = xCenter + halfWidth - 3;
    rect.top    = top;
    rect.bottom = top + 4;

    HBRUSH hbr = CreateSolidBrush( backgroundColor );
    FillRect( hDC, &rect, hbr );
    DeleteBrush( hbr );

    // draw the focus rectangle

    DrawFlash( hWnd, hDC, GetFlashState( hWnd ), TRUE );

    // clean up and exit

    SelectObject( hDC, oldPen );

    DeletePen( hBorderPen );
    DeletePen( hHighlightPen );
    DeletePen( hShadowPen );

    EndPaint( hWnd, &paintStruct );
}

//-----------------------------------------------------------
// we define an explicit WM_ERASEBKGND handler to avoid providing an
// hBrush in the WNDCLASS definition.  BoundsChecker claims it does not
// get deleted.  Whether true or not, it is comforting to not see it
// in the debug log.
BOOL DgnSlideControl::OnEraseBkGnd( HWND hWnd, HDC hDC )
{
    RECT rect;

    GetClientRect(hWnd, &rect);
    HBRUSH hbr = CreateSolidBrush( backgroundColor );
    FillRect( hDC, &rect, hbr );
    DeleteBrush( hbr );

    return TRUE;
}

////////////////////////////////////////////////////////////////////////

//----------------------------------------------------------------------

extern "C"
LRESULT _export FAR PASCAL DgnSlideControlWndProc( HWND hWnd,
                                                   UINT message, WPARAM wParam,
  => LPARAM lParam )
{
    return slideControl.wndProc( hWnd, message, wParam, lParam );
}

//----------------------------------------------------------------------

void DgnSlideControl::libMainInit( )
{
```

SLIDECTL.CPP 7-22-95 11:29a

```
    WNDCLASS wndClass;
    memset( &wndClass, '\0', sizeof(wndClass) );

    // We are not based on any existing class.

    m_myDefProc = DefWindowProc;
    m_myOffset  = 0;

    wndClass.style = CS_GLOBALCLASS | CS_DBLCLKS;
    wndClass.lpfnWndProc   = DgnSlideControlWndProc;
    wndClass.cbWndExtra    = m_myOffset + 5*sizeof(int);
    wndClass.hInstance     = hInstance;
    wndClass.lpszClassName = DGN_SLIDECONTROL;

    // try to register the class; if it fails the first time, delete
    // the existing class and try again

    if( !::RegisterClass( &wndClass ) ) {

        WNDCLASS existingWndClass;
        GetClassInfo( 0, DGN_SLIDECONTROL, &existingWndClass );
        UnregisterClass( DGN_SLIDECONTROL, existingWndClass.hInstance );

        ::RegisterClass( &wndClass );
    }
}
```

Page 7 of 7

```
/*************************************************************************
        Description

        statdlg.cpp
        Also memory computation

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995

*************************************************************************/

#include <toolhelp.h>
#include <string.h>

const char *moduleList[] =
{
    "WAPP",
    "DRAGDEV",
    "DRAGACPA",
    "DRAGMEDI",
    "DRAGON",
    0
};

// Compute the memory consumption by walking the global heap.
// Count every non-discardable block which belongs to our EXE or DLLs.

long computeMemoryUse()
{
    long totalMemory = 0;

    // load global information
    GLOBALINFO  gInfo;
    gInfo.dwSize = sizeof( GLOBALINFO );

    GlobalInfo( &gInfo );

    // loop through the global heap
    GLOBALENTRY gEntry;
    gEntry.dwSize = sizeof( GLOBALENTRY );

    MODULEENTRY mEntry;
    mEntry.dwSize = sizeof( MODULEENTRY );

    TASKENTRY taskEntry;
    taskEntry.dwSize = sizeof( TASKENTRY );

    for( BOOL bHasEntry = GlobalFirst( &gEntry, GLOBAL_ALL );
         bHasEntry;
         bHasEntry = GlobalNext( &gEntry, GLOBAL_ALL ) )
    {
        // ignore free blocks
        if( !gEntry.hOwner )    continue;

        // ignore the following types of data blocks
        switch( gEntry.wType )
        {
         case GT_CODE:
         case GT_FREE:
         case GT_INTERNAL:
         case GT_SENTINEL:
         case GT_BURGERMASTER:
            continue;
```

STATDLG.CPP 7-22-95 11:31a

```
         default:
            break;
        }

        // get the module name; if this fails, try to get the task name
        if( ModuleFindHandle( &mEntry, (HMODULE )gEntry.hOwner ) )
            continue;

        if( TaskFindHandle( &taskEntry, (HTASK )gEntry.hOwner ) )
            strcpy( mEntry.szModule, taskEntry.szModule );

        for( char** module = moduleList; *module; ++module )
        {
            if( stricmp( *module, mEntry.szModule) )
            {
                totalMemory += gEntry.dwBlockSize;
                break;
            }
        }
    }

    return totalMemory;
}
```

Page 1 of 1

```
////////////////////////////////////////////////////////////////////////////
/*

    Description

        WCOACH.CPP
        Example of modification to WAPP.CPP. It is a training application
        based on the code in WAPP.CPP

        Main module: Windows loop and algoritm flow in onSpeech()

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995

*/
        FUNCTIONS

#include <windowsx.h>
#include <string.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <bios.h>
#include <io.h>
#include <fcntl.h>
#include "dragcpp.h"

extern "C" {
    #include "fepext.h"
}

#include "wcoach.h"
#include "strhash.h"
#include "slidectl.h"


#include <mmsystem.h>
#include <ddeml.h>


// MessageBox( DlgWin::hWnd, "In onDestroy", "not dead yet?", MB_OK | MB_ICONEXC
  => LAMATION );

// This is the instance of the DgnSlideControl
DgnSlideControl slideControl;
extern HINSTANCE hinstance;      // when we call DgnSlideControl::libMainInit()
HINSTANCE hinstance;             // hinstance must be initialized...

#include "hashcaz.h"
#include "deamon.h"

#define WAKEUP_STATE    0
#define ZIP_STATE       1
#define CITY_STATE      2

#define handleMessage( message, hWnd, wParam, lParam, fn ) \
    case (message) : returnValue = HANDLE_ ## message( (hWnd), (wParam), (lParam
  => ), (fn) ); break;


static char femCityVoc[]        = "ctall.vof";
static char femCityDigitsVoc[]  = "cdigits.vof";
```

WCOACH.CPP 7-22-95 11:33a

```
static char femDigitsVoc[]      = "digits.vof";
static char malCityVoc[]        = "ctall.vom";
static char malCityDigitsVoc[]  = "cdigits.vom";
static char malDigitsVoc[]      = "digits.vom";
static char* cityVoc = 0;
static char* cityDigitsVoc = 0;
static char* digitsVoc = 0;
static char cityUser[ 128 ];
static char digitsUser[ 128 ];

static char userPathString[ 256 ] = ".";

static char* sayPrompt[] = {
    "Say: WakeUp",
    "Say: ZipState",
    "Say: City",
};


class ChoiceListStats
{
 protected:

    int digitStats[ 500 ];

    char openRecResult[ 128 ];

 public:

    void setOpenRecResult( const char* src ) { strcpy( openRecResult, src ); }

    void recordResult( const char* src );

    void printStats();
};

void ChoiceListStats::recordResult( const char* src )
{
    char *o = openRecResult;

    for( int i = 0; i < 5; ++i )
    {
        if( !isdigit( *o ) || !isdigit( *src ) )
            return;

        digitStats[ i*100 + 10*( *src - '0' ) + ( *o - '0' ) ] += 1;

        if( *++o && *++src == ' ' )
            ++src;

        else
            return;
    }
}

void ChoiceListStats::printStats()
{
    FILE *out = fopen( "digstats", "a+" );

    if( out )
    {
        int *v = digitStats;

        fprintf( out, "////////////////////////////////////////\n" );

        for( int i=0; i<5; ++i )
```

Page 1 of 19

```
        {
            fprintf( out, "// %d\n", i );

            for( int j=0; j<10; ++j )
            {
                fprintf( out, "%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t\n",
                        v[0], v[1], v[2], v[3], v[4],
                        v[5], v[6], v[7], v[8], v[9] );
                v += 10;
            }
        }

        fclose( out );

        memset( openRecResult, 0, sizeof( openRecResult ) );
    }
}

static ChoiceListStats choiceListStats;



class DlgWin
{
 public:
    static long FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );
    static HWND  hWnd;

 protected:

    static bool messageHandled;
    static bool sawError;
    static bool ignoreErrors;
    static char* name;
    static short wordPause;

    static SpeechTask *speechTask;              // speech driver task ID
    static UttChannel *channel;
    static Recognizer* recog;
    static UttChannel zipUtt, cityUtt;

    static OC< char > cityHypo;
    static AC< SD_WORD > cityIds;
    static AC< long > zips;
    static long zipValue;
    static char zipPhrase[ UTT_PROMPT_LENGTH ];
    static char zipSpelling[ UTT_PROMPT_LENGTH ];
    static char digitSpelling[ UTT_PROMPT_LENGTH ];
    static char stateSpelling[ UTT_PROMPT_LENGTH ];
    static char *zipState;
    static bool zipAgreesWithState;
    static bool fixError;
    static long numErrors;
    static bool wasListening;

    static int appState;                        // zips, city, state, error
    static SD_WORD cancelWord;
    static SD_WORD badLabelWord;
    static SD_WORD goToSleepWord;
    static SD_WORD wakeUpWord;

    static char buf[ UTT_PROMPT_LENGTH ];       // scratch buf
    static Comm* comm[4];

    static void reportError( int code, char far *message );
    static void SD_CALLBACK _export postSpeechEvent( SD_CHANNEL, SD_CHANNEL_EVEN
```

```
=> T );

    // wndProc() services
    static bool onInitDialog( HWND hWnd, HWND topCntrl, long );
    static void onDestroy( HWND hWnd );
    static void onSysCommand( HWND hWnd, UINT cmd, int x, int y );
    static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y );
    static void onSpeech( HWND hWnd, SD_CHANNEL ch );
    static void onCommCheckMsg( HWND hWnd, Comm* cm );
    static void onCommSendSyncMsg( HWND hWnd, Comm* cm );
    static void onTimerDrawVU( HWND hwnd, UINT id );

    // sub service routines for the main winProc services
    static void setDigitsUser( char* usrName );
    static bool isGoodZipPhrase();
    static bool isCancelBadLabelGoToSleep( HWND );
    static void itsGoodAdapt();
    static void itsBad( HWND, UttChannel*  );
    static bool DlgWin::rerecogZip( long* foundZip, int* foundDistance );
    static void outCommPort();

    friend int PASCAL WinMain( HINSTANCE, HINSTANCE, LPSTR, int );
};

// DlgWin Globals...

HWND            DlgWin::hWnd;
bool            DlgWin::messageHandled;
SpeechTask*     DlgWin::speechTask;
Recognizer*     DlgWin::recog;
UttChannel*     DlgWin::channel;
bool            DlgWin::sawError;
bool            DlgWin::ignoreErrors;
char*           DlgWin::name = "dragon123\0      ";

UttChannel      DlgWin::zipUtt;
UttChannel      DlgWin::cityUtt;

OC< char >      DlgWin::cityHypo;
AC< SD_WORD >   DlgWin::cityIds;
AC< long >      DlgWin::zips;
long            DlgWin::zipValue;
char            DlgWin::zipPhrase[ UTT_PROMPT_LENGTH ];
char            DlgWin::zipSpelling[ UTT_PROMPT_LENGTH ];
char            DlgWin::digitSpelling[ UTT_PROMPT_LENGTH ];
char            DlgWin::stateSpelling[ UTT_PROMPT_LENGTH ];
char*           DlgWin::zipState;
bool            DlgWin::zipAgreesWithState;
bool            DlgWin::wasListening;
bool            DlgWin::fixError = 1;
long            DlgWin::numErrors;
short           DlgWin::wordPause = 256;

int             DlgWin::appState;                       // zips, city, state, er
  => ror
SD_WORD         DlgWin::cancelWord = 0;
SD_WORD         DlgWin::badLabelWord = 0;
SD_WORD         DlgWin::goToSleepWord = 0;
SD_WORD         DlgWin::wakeUpWord = 0;


char            DlgWin::buf[ UTT_PROMPT_LENGTH ];       // scratch buf
Comm*           DlgWin::comm[4];
```

```
////////////////////////////////////////////////////////////////////////
// configuration file stuff
////////////////////////////////////////////////////////////////////////


struct ConfigCommands
{
    const char* name;
    char* (*command)( char* );
};


static char* setCommand( char* );

static ConfigCommands cfgCommands[] = {
    { "set", setCommand },
    { 0, 0 }
};


////////////////////////////////////////////////////////////////////////
// set a variable... the particular data and functions known to setCommand()

static CitySL badLabel( 1, "badlabel" );
static CitySL firstDayAir( 2, "firstdayair" );
static CitySL nextDayAir( 3, "nextdayair" );

static CitySL alberta( 20, "alberta" );
static CitySL britishColumbia( 21, "britishcolumbia" );
static CitySL labrador( 22, "labrador" );
static CitySL manitoba( 23, "manitoba" );
static CitySL newBrunswick( 24, "newbrunswick" );
static CitySL newfoundland( 25, "newfoundland" );
static CitySL northWestTerritories( 26, "northwestterritories" );
static CitySL novaScotia( 27, "novascotia" );
static CitySL ontario( 28, "ontario" );
static CitySL princeEdwardIsland( 29, "princeedwardisland" );
static CitySL quebec( 30, "quebec" );
static CitySL saskatchewan( 31, "saskatchewan" );
static CitySL yukonTerritory( 32, "yukonterritory" );

static CitySL mexico( 40, "mexico" );


static char* setSpecialZip( CitySL* sl, char* value )
{
    int i = 0;

    for( char *s=value; *s; ++s )
    {
        i += ( *s >= '0' && *s <= '9' );
    }

    if( i == 0 )
        return "Bad special Zip code assingment";

    long test = atol( value );

    sl->zip = test;

    return 0;
}
```

MCDACH.CPP 7-22-95 11:33a

```
static char* setBadLabelZip( char* value )                          { return setSpecialZip(
  => &badLabel, value ); }
static char* setFirstDayAirZip( char* value )                       { return setSpecialZip(
  => &firstDayAir, value ); }
static char* setNextDayAirZip( char* value )                        { return setSpecialZip(
  => &nextDayAir, value ); }
static char* setAlbertaZip( char* value )                           { return setSpecialZip(
  => &alberta, value ); }
static char* setBritishColumbiaZip( char* value )                   { return setSpecialZip(
  => &britishColumbia, value ); }
static char* setLabradorZip( char* value )                          { return setSpecialZip(
  => &labrador, value ); }
static char* setManitobaZip( char* value )                          { return setSpecialZip(
  => &manitoba, value ); }
static char* setNewBrunswickZip( char* value )                      { return setSpecialZip(
  => &newBrunswick, value ); }
static char* setNewFoundlandZip( char* value )                      { return setSpecialZip(
  => &newFoundland, value ); }
static char* setNorthWestTerritoriesZip( char* value )              { return setSpecialZip(
  => &northWestTerritories, value ); }
static char* setNovaScotiaZip( char* value )                        { return setSpecialZip(
  => &novaScotia, value ); }
static char* setOntarioZip( char* value )                           { return setSpecialZip(
  => &ontario, value ); }
static char* setPrinceEdwardIslandZip( char* value )                { return setSpecialZip(
  => &princeEdwardIsland, value ); }
static char* setQuebecZip( char* value )                            { return setSpecialZip(
  => &quebec, value ); }
static char* setSaskatchewanZip( char* value )                      { return setSpecialZip(
  => &saskatchewan, value ); }
static char* setYukonTerritoryZip( char* value )                    { return setSpecialZip(
  => &yukonTerritory, value ); }
static char* setMexicoZip( char* value )                            { return setSpecialZip(
  => &mexico, value ); }



static CommPars* comPars[ 4 ];
static char* commformat[ 4 ];

static char* setDefaultCommPars( char* value )
{
    for( char *s = value; *s; ++s )
        if( *s == ',' )   *s = ' ';

    char parityStr[ 10 ], comPort[ 20 ], formatString[ 128 ];

    int baud, dataBits, stopBits;

    int numFields = sscanf( value, "%s %d %s %d %d %s",
                            comPort, &baud, parityStr, &dataBits, &stopBits, for
  => matString );
    if( numFields < 5 )
        return "wrong number/order of arguments... ex:\n\tset comm = com1 9600 n
  =>  8 1";

    // check to see if the arguments are reasonable and set the port...

    int port=comPort[ 3 ] - '0';

    if( 0>port || port>4 )
        return "bad commPort number, (must be 1 thru 4)";

    if( baud != 19200 && baud != 14400 && baud != 9600 &&
```

Page 3 of 19

```
        baud != 4800 && baud != 2400 && baud != 1200 )
        return "unknown commPort Baud rate";

    int parity;

    switch( tolower( *parityStr ) )
    {
     case 'n':  parity = NOPARITY;       break;
     case 'o':  parity = ODDPARITY;      break;
     case 'e':  parity = EVENPARITY;     break;
     case 'm':  parity = MARKPARITY;     break;
     case 's':  parity = SPACEPARITY;    break;

     default:
        return "unknown commPort Parity";
    }

    if( dataBits != 7 && dataBits != 8 )
        return "strange number of commPort dataBits";

    if( stopBits == 1 )
        stopBits = ONESTOPBIT;

    else if( stopBits == 2 )
        stopBits = TWOSTOPBITS;

    else
        return "strange number of commPort stopBits";

    if( numfields == 5 )
        commFormat( port-1 ) = 0;

    else
    {
        commFormat( port-1 ) = new char[ strlen( formatString ) + 1 ];

        strcpy( commFormat[ port-1 ], formatString );
    }

    comPars[ port-1 ] = new CommPars( port, baud, parity, dataBits, stopBits );

    return 0;
}

static char* setPureState( char* value )
{
    for( char *s = value; *s && !isspace( *s ); ++s );

    *s = '\0';

    StateZip* sz = StateZip::findStateZip( value );

    if( sz == 0 )
        return "bad Pure State in UPS.CFG";

    sz->pure( TRUE );

    return 0;
}
static char* setUserPath( char* value )
{
    for( char *s = value; *s && !isspace( *s ); ++s );

    *s = '\0';
```

```
    strcpy( userPathString, value );

    return 0;
}

static char* setFepPar( WORD id, char* value )
{
    for( char *s = value; *s && !isspace( *s ); ++s );

    *s = '\0';

    if( *value )
    {
        WORD v;

        v = (WORD )atol( value );

        Fep_Setpar( SUPER_USER, id, v );

        return 0;
    }
    return "Par Not Set";
}

static char* setStartOfSpeech( char* value )
{
    if( setFepPar( 13, value ) )
        return "StartOfSpeech Not Set";

    return 0;
}

static char* setEndOfSpeech( char* value )
{
    if( setFepPar( 12, value ) )
        return "EndOfSpeech Not Set";

    return 0;
}


static ZipRangeOC zroc;

static char* setZipRange( char* value )
{
    for( char *s = value; *s; ++s )
        if( *s == ',' )  *s = ' ';

    long lowZip, highZip;

    char preName[ 20 ], postName[ 20 ];

    int numfields = sscanf( value, "%ld %ld %s %s",
                            &lowZip, &highZip, preName, postName );

    if( numfields == 3 )
        zroc.addRange( lowZip, highZip, preName, 0 );

    else if( numfields == 4 )
        zroc.addRange( lowZip, highZip, preName, postName );

    else
        return "wrong number arguments... ex:\n\tset ZipRange = 92262 92262 belt
»  25";
```

```
    return 0;
}

static char* setNothing( char* )
{
    return 0;
}

static ConfigCommands setCommands[] = {
    { "Com",                    setDefaultCommPars },
    { "Comm",                   setDefaultCommPars },
    { "ComPort",                setDefaultCommPars },
    { "Pure",                   setPureState },
    { "UserPath",               setUserPath },
    { "StartOfSpeech",          setStartOfSpeech },
    { "EndOfSpeech",            setEndOfSpeech },

    { "ZipRange",               setZipRange },

    { "BadLabel",               setBadLabelZip },
    { "FirstDayAir",            setFirstDayAirZip },
    { "NextDayAir",             setNextDayAirZip },

    { "Alberta",                setAlbertaZip },
    { "BritishColumbia",        setBritishColumbiaZip },
    { "Labrador",               setLabradorZip },
    { "Manitoba",               setManitobaZip },
    { "NewBrunswick",           setNewBrunswickZip },
    { "NewFoundland",           setNewFoundlandZip },
    { "NorthWestTerritories",   setNorthWestTerritoriesZip },
    { "NovaScotia",             setNovaScotiaZip },
    { "Ontario",                setOntarioZip },
    { "PrinceEdwardIsland",     setPrinceEdwardIslandZip },
    { "Quebec",                 setQuebecZip },
    { "Saskatchewan",           setSaskatchewanZip },
    { "YukonTerritory",         setYukonTerritoryZip },

    { "Mexico",                 setMexicoZip },

    { 0, 0 }
};

static char* setCommand( char* var )
{
    while( isspace( *var ) )
        ++var;

    char* value = var;

    while( isalnum( *value ) || *value == '_' )
        ++value;

    if( *value == '\0' )
        return "no value found in set command... ex: set variable = value";

    bool foundEqual = (*value == '=');

    *value++ = '\0';

    if( !foundEqual )
    {
        while( isspace( *value ) )
            ++value;

        if( *value != '=' )
```

WCDACH.CPP 7-22-95 11:33a

```
            return "no \"=\" sign in set command... ex: set variable = value";

        ++value;
    }

    while( *value && isspace( *value ) )
        ++value;

    for( int i=0; setCommands[ i ].name != 0; ++i )
    {
        if( stricmp( var, setCommands[ i ].name ) == 0 )
            return setCommands[ i ].command( value );
    }

    return "unknown set variable";
}

void parseConfig( const char* filename )
{
    FILE *cfgFile = fopen( filename, "r" );

    char *errorMessage = ( cfgFile == 0
                            ? "could not open UPS.CFG file"
                            : 0 );

    char line[ 256 ];
    int lineNumber = 0;

    while( errorMessage == 0 && fgets( line, sizeof( line ), cfgFile ) != 0 )
    {
        char *var = line;

        ++lineNumber;

        while( isspace( *var ) )
            ++var;

        if( *var == '#' || *var == '\0' )
            continue;

        char *value = var;

        while( !isspace( *value ) && *value != '\0' )
            ++value;

        if( *value != '\0' )
        {
            *value++ = '\0';

            while( isspace( *value ) )
                ++value;
        }

        for( int i=0; cfgCommands[ i ].name != 0; ++i )
        {
            if( stricmp( var, cfgCommands[ i ].name ) == 0 )
            {
                errorMessage = cfgCommands[ i ].command( value );

                break;
            }
        }
    }

    if( errorMessage )
    {
```

Page 5 of 19

```
        char errorType[ 40 ] = "UPS.CFG error, line          \n";

        itoa( lineNumber, errorType + 20, 10 );

        MessageBox( DlgWin::hWnd, errorMessage, errorType, MB_OK | MB_ICONEXCLAM
  => ATION );
    }

    if( cfgFile )
        fclose( cfgFile );
}

//////////////////////////////////////////////////////////////////////////////
// onInitDialog
//////////////////////////////////////////////////////////////////////////////

class FindFilePath
{
 protected:

    static char defaultPath[];
    static char defaultFileSpec[];

    char *pathString, *path;
    char *fileSpec;
    int attribFlag;

    struct ffblk attrib;

    bool findFirstPrimitive();

 public:

    FindFilePath( char* fspec, char* p =0, int flag =FA_NORMAL );

    ~FindFilePath()
    {
        if( pathString != defaultPath )
            delete pathString;

        if( fileSpec != defaultFileSpec )
            delete fileSpec;
    }

    bool FindFilePath::findFirst();
    bool FindFilePath::findNext();

    char* filename() { return attrib.ff_name; }
};

char FindFilePath::defaultFileSpec[] = "*.*";
char FindFilePath::defaultPath[] = ".\0";

FindFilePath::FindFilePath( char* fspec, char* p, int flag )
{
    attribFlag = flag;

    if( p )
    {
        path = pathString = new char[ strlen( p ) + 2 ];
        strcpy( pathString, p );
        for( p = pathString; *p; ++p )
            if( *p == ';' )
                *p = '\0';
```

WCOACH.CPP 7-22-95 11:33a

```
        *++p = '\0';
    }
    else
        pathString = path = defaultPath;

    if( fspec )
    {
        fileSpec = new char[ strlen( fspec ) + 1 ];
        strcpy( fileSpec, fspec );
    }
    else
        fspec = defaultFileSpec;
}

bool FindFilePath::findFirstPrimitive()
{
    for(;;)
    {
        if( *path == '\0' )
            return FALSE;

        char* fSpec = new char[ strlen( path ) + strlen( fileSpec ) + 2 ];

        char* p = stpcpy( fSpec, path );

        if( p>fSpec && *(p-1) != ':' && *(p-1) != '\\' && *(p-1) != '/' )
            *p++ = '/';

        p = stpcpy( p, fileSpec );

        path +=  + strlen( path ) + 1;

        if( !findfirst( fSpec, &attrib, attribFlag ) )
        {
            delete fSpec;
            return TRUE;
        }
        delete fSpec;
    }
}

bool FindFilePath::findFirst()
{
    path = pathString;

    return findFirstPrimitive();
}

bool FindFilePath::findNext()
{
    if( !findnext( &attrib ) )
        return TRUE;

    return findFirstPrimitive();
}

bool DlgWin::onInitDialog( HWND hDlg, HWND, long )
{
    hWnd = hDlg;

    speechTask = new SpeechTask( name, reportError );
```

Page 6 of 19

```
    recog = 0;

    StateZip::initZipToUsaStateZip();

//    fep_Setpar( SUPER_USER, 13, 87 );
//    fep_Setpar( SUPER_USER, 12, 58 );

    parseConfig( "ups.cfg" );

    for( int i=0; i<4; ++i )
    {
        if( comPars[ i ] != 0 )
        {
            comm[ i ] = new Comm( hDlg, comPars[ i ] );

            if( comm[ i ]->error() != 0 )
            {
                MessageBox( hDlg, "Comm Error", "Port Initialization Failed", MB
 => _OK | MB_ICONEXCLAMATION );

                delete comm[ i ];

                comm[ i ] = 0;
            }

            delete comPars[ i ];
        }
    }

    FindFilePath ffPath( "*.un?", userPathString );

    // set up the usr listbox
    if( ffPath.findFirst() )
    {
        do
        {
            SendDlgItemMessage( hDlg, USR_COMBO_ID, CB_ADDSTRING, 0, (DWORD) ffP
 => ath.filename() );
        }
        while( ffPath.findNext() );
    }


    appState = ZIP_STATE;

//    speechTask->setPar( "computation", (short )12 );
//    speechTask->setPar( "adapt-tolerance", (short )3 );

    SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLRANGE, 0, 0x02000
 => 064L );

    SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLPOS, 0x0100, 1 );


    for( i=0; i<100; ++i )
    {
        char zipFileName[] = "zip/zip.000000000";

        itoa( i, zipFileName + 8 + (i<10), 10 );

        FILE* file = fopen( zipFileName, "r" );

        if( file == 0 )     continue;

        fclose( file );
```

```
        StateZip::initHSs( zipFileName );
    }

    return TRUE;
}


void formatOutput( char* outBuf, const char* format,
                   const char* zip, const char* state, const char* city )
{
    char* buf = outBuf;

    while( *format )
    {
        switch( *format )
        {
         case '%':
            ++format;
            if( tolower( *format ) == 'z' )
            {
                const char* z=zip;
                while( (*buf = *z++) != 0 )
                    ++buf;

                ++format;
            }
            else if( tolower( *format ) == 's' )
            {
                const char* s=state;
                while( (*buf = *s++) != 0 )
                    ++buf;

                ++format;
            }
            else if( tolower( *format ) == 'c' )
            {
                const char* c=city;
                while( (*buf = *c++) != 0 )
                    ++buf;

                ++format;
            }
            else
                *buf++ = '%';
            break;

        case '\\':
            ++format;

            if( tolower( *format ) == 'r' )
            {
                *buf++ = '\r';

                ++format;
            }
            else if( tolower( *format ) == 'n' )
            {
                *buf++ = '\n';

                ++format;
            }
            else if( tolower( *format ) == 'b' )
            {
                *buf++ = ' ';

                ++format;
            }
            else if( tolower( *format ) == 't' )
            {
                *buf++ = '\t';

                ++format;
            }
            else
            {
                if( *format != '\0' )
                {
                    *buf++ = *format++;
                }
            }
            break;

        default:
            *buf++ = *format++;
        }
    }

    *buf = 0;

    assert( buf - outBuf < 256 );
}



void DigWin::outCommPort()
{
    wasListening = channel->isListening();

    zroc.playWave( atol( digitSpelling ) );
    //channel->beep( "ding.wav", 0 );

    char outBuf[ 256 ];

    for( int i=0; i<4; ++i )
    {
        if( comm[ i ] )
        {
            formatOutput( outBuf, commformat[ i ] ? commformat[ i ] : "%z\r",
                        digitSpelling, stateSpelling, cityUtt.prompt() );

            comm[ i ]->comWrite( outBuf, strlen( outBuf ) );
        }
    }

    static long lastInterval = 0;
    static time_t lastTime = 0;

    long currentTime = biostime( 0, 0 );

    lastInterval = (lastInterval + (currentTime - lastTime)) / 2;

    if( lastInterval > 1000 )
        lastInterval = 1000;

    long rate = (3600L * 1000L / 55L) / lastInterval;

    lastTime = currentTime;

    ltoa( rate, outBuf, 10 );
```

WCOACH.CPP 7-22-95 11:33a

```
    char *s = outBuf + strlen( outBuf );

    strcpy( s, "  pkgs/hour       " );

    SetWindowText( GetDlgItem( hWnd, STAT1_TEXT_ID ), outBuf );

    static long totalNumberOfPackages = 0;

    ltoa( ++totalNumberOfPackages, outBuf, 10 );

    s = outBuf + strlen( outBuf );

    strcpy( s, "  pkgs       " );

    SetWindowText( GetDlgItem( hWnd, STAT2_TEXT_ID ), outBuf );

    ltoa( numErrors, outBuf, 10 );

    s = outBuf + strlen( outBuf );

    strcpy( s, "  BadLabels       " );

    SetWindowText( GetDlgItem( hWnd, STAT3_TEXT_ID ), outBuf );
}


void DigWin::reportError( int code, char far *message )
{
    sawError = YES;
    if( ignoreErrors )
        return;          // prevent recursion
    ignoreErrors = YES;

    // Throw up an error message
    char buffer[ 512 ];
    sprintf( buffer, "An error has occurred\n  code = %d\n  message = %s",
            code, message );

    if( stricmp( message, "Bad token" ) )
    {
        MessageBox( hWnd, buffer, "WinTalk SDAPI Error",
                    MB_TASKMODAL | MB_ICONSTOP | MB_OK );

        // terminate...  kill the hTopParentWindow
        if( hWnd )
            PostMessage( hWnd, WM_CLOSE, 0, 0 );
    }
    ignoreErrors = 0;
}

void SD_CALLBACK _export DigWin::postSpeechEvent( SD_CHANNEL ch, SD_CHANNEL_EVEN
=> T eventIn )
{
    assert( eventIn == SD_CHANNEL_START );

    MSG msg;

    if( 0 == PeekMessage( &msg, hWnd, UM_CHANNELSTART, UM_CHANNELSTART, PM_NOREM
=> OVE | PM_NOYIELD ) )
        PostMessage( hWnd, UM_CHANNELSTART, (WORD )ch, 0 );

} // EventHandler
```

Page 8 of 19

```
#define HANDLE_UM_CHANNELSTART(hwnd, wParam, lParam, fn) \
    ((fn)(hwnd, (SD_CHANNEL )wParam), 1L)


#define HANDLE_UM_COMM_CHECK_MSG( hwnd, wParam, lParam, fn ) \
    ((fn)(hwnd, (Comm* )lParam), 1L)


#define HANDLE_UM_COMM_SEND_SYNC_MSG( hwnd, wParam, lParam, fn ) \
    ((fn)(hwnd, (Comm* )lParam), 1L)



void DigWin::onCommCheckMsg( HWND, Comm* cm )
{
    assert( cm != 0 );
    assert( cm == comm[0] || cm == comm[1] || cm == comm[2] || cm == comm[3] );

    cm->checkMessage();
}


void DigWin::onCommSendSyncMsg( HWND, Comm* cm )
{
    assert( cm != 0 );
    assert( cm == comm[0] || cm == comm[1] || cm == comm[2] || cm == comm[3] );

    cm->sendSyncMsg();
}


int DigWin::isGoodZipPhrase()
{
    char *zph = zipPhrase;
    char *zsp = zipSpelling;
    char *ssp = stateSpelling;
    char *dsp = digitSpelling;

    for( int len = 0; *zph != '\0'; )
    {
        if( isdigit( *zph ) )
        {
            *dsp++ = *zsp++ = *zph++;
            ++len;
        }

        else if( isspace( *zph ) )
            ++zph;

        else
        {
            if( zsp != zipSpelling )            *zsp++ = ' ';

            while( (*ssp++ = *zsp++ = *zph++) != '\0' )
                ;
            break;
        }
    }

    *zsp = *ssp = *dsp = '\0';

    choiceListStats.setOpenRecResult( digitSpelling );

    return( len + (ssp != stateSpelling) );
}
```

```
static long right=0, wrong=0, predicted=0, addressConfidence=100;


class ZipHypothesis
{
 public:
    bool isNewWord;
    SD_WORD word;
    long value;

    char spelling[ UIT_PROMPT_LENGTH ];
    char phrase[ UIT_PROMPT_LENGTH ];

    ZipHypothesis( Recognizer*, long zip, char* stateName );
    ZipHypothesis( Recognizer*, const char*, const char* );
};

class ZipHypoAC : public AC< ZipHypothesis >
{
 public:
    void removeAll( Recognizer* );
};

void ZipHypoAC::removeAll( Recognizer* recog )
{
    for( int i=count(); i--; )
    {
        if( (*this)[ i ].isNewWord )
            recog->deleteWord( (*this)[ i ].word );
    }
    AC< ZipHypothesis >::removeAll();
}

const char* ltoZip( long num )
{
    static char buf[ UIT_PROMPT_LENGTH ] = "00000";

    ltoa( num, buf+5, 10 );

    return buf + strlen( buf ) - 5;
}


ZipHypothesis::ZipHypothesis( Recognizer* recog, long num, char* stateName )
{
    value = num;

    strcpy( spelling, ltoZip( num ) );

    char *p = phrase;
    char *s = spelling;

    while( (*p = *s) != '\0' )
    {
        ++s;            ++p;            *p++ = ' ';
    }

    *s++ = *p++ = ' ';

    strcpy( s, stateName );

    strcpy( p, stateName );

    isNewWord = ( (word=recog->wordId( spelling )) == 0 );
    if( isNewWord )
        word = recog->buildWord( spelling, phrase );

    else
        recog->addWord( word );

    assert( word != 0 );
}

ZipHypothesis::ZipHypothesis( Recognizer* recog, const char* zSpell, const char*
  =>  zPhrase )
{
    value = atol( zSpell );

    strcpy( spelling, zSpell );

    strcpy( phrase, zPhrase );

    isNewWord = ( (word=recog->wordId( spelling )) == 0 );

    if( isNewWord )
        word = recog->buildWord( spelling, phrase );

    else
        recog->addWord( word );

    assert( word != 0 );
}


bool DlgWin::rerecogZip( long* foundZip, int* foundDistance )
{
    static SD_STATE tmpState=0;

    static ZipHypoAC zipHypoAC;

    *foundZip = -1;

    tmpState = recog->newState( "tmpzip" );

    recog->setState( tmpState );

    StateZip* sz = StateZip::findStateZip( stateSpelling );

    if( sz )
    {
        cityUtt.resultName( buf, sizeof(buf) );
        // could do more, resultName( 1 )...  watch out for canceladdress...

        zips.removeAll();

        sz->cityToZips( buf, &zips );

        int imax=zips.count() + 1;

        zipHypoAC.makeRoom( imax );

        new( &(zipHypoAC[0]) )  ZipHypothesis( recog, zipSpelling, zipPhrase );

        for( int i=1; i<imax; ++i )
        {
            new( &(zipHypoAC[i]) )  ZipHypothesis( recog, zips[ i-1 ], stateSpel
  => ling );
        }
```

WCOACH.CPP  7-22-95  11:33a

```
        UttChannel *ch = recog->getChannel();

        recog->setChannel( &zipUtt );

        if( recog->recog() == 0 )
        {
            if( zipUtt.resultCount() > 1 )
            {
                SD_WORD w = zipUtt.resultId();

                int i = (w == zipHypoAC[0].word);

                w = zipUtt.resultId( i );

                *foundDistance = zipUtt.resultDistance( i );

                for( i=0; i<imax; ++i )
                {
                    if( zipHypoAC[ i ].word == w )
                    {
                        *foundZip = zipHypoAC[ i ].value;
                        break;
                    }
                }
            }
        }

        zipHypoAC.removeAll( recog );

        recog->setChannel( ch );
    }

    recog->deleteState( tmpState );

    return( *foundZip != -1 );
}

bool DlgWin::isCancelBadLabelGoToSleep( HWND hwnd )
{
    if( recog->resultId() == cancelWord )
    {
        recog->beep( "canceled.wav" );

        appState = ZIP_STATE;
    }
    else if( recog->resultId() == badLabelWord )
    {
        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), "BadLabel" );

        strcpy( digitSpelling, ltoZip( badLabel.zip ) );

        numErrors++;

        outCommPort();

        appState = ZIP_STATE;
    }
    else if( recog->resultId() == goToSleepWord )
    {
        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), "Sleeping" );

        recog->beep( "sleeping.wav" );

        appState = WAKEUP_STATE;
    }
    else
```

Page 10 of 19

```
        return 0;

    SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_ID ), "" );

    SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_ID ), "" );

    recog->killUtt();

    return 1;
}

void DlgWin::itsBad( HWND hwnd, UttChannel* toMe )
{
    if( toMe == &cityUtt  && !fixError )
    {
        cityUtt.setPrompt( "BadLabel" );

        strcpy( digitSpelling, itoZip( badLabel.zip ) );

        strcpy( stateSpelling, "BadLabel" );

        numErrors++;

        outCommPort();

        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), "BadLabel" );
    }
    else
    {
        recog->beep( "bad.wav" );

        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), "no recognition" );
    }

    SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_ID ), "" );

    SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_ID ), "" );

    recog->giveAwayUtt( toMe );

    toMe->setPrompt( "" );

    appState = ZIP_STATE;
}

void DlgWin::itsGoodAdapt()
{
    choiceListStats.recordResult( zipUtt.prompt() );

    // lets adapt...
    recog->setUser( cityUser );
    recog->setVoc( cityVoc );
    recog->setChannel( &cityUtt );
    recog->adapt();

    if( zipAgreesWithState )
    {
        recog->setChannel( &zipUtt );

        recog->setVoc( cityDigitsVoc );
        recog->contAdapt();

        recog->setUser( digitsUser );
        recog->setVoc( digitsVoc );
        recog->contAdapt();
    }
```

```
    recog->setChannel( channel );

    appState = ZIP_STATE;
}

void DlgWin::onSpeech( HWND hwnd, SD_CHANNEL ch )
{
    // get the utterance handle; do not wait for the end of the
    // utterance since the recognition can proceed in parallel
    // with the utterance collection

    assert( recog );

    if( recog->notifyChannel( ch, SD_CHANNEL_START ) )
    {
        SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_RESETCONTENT, 0, 0 );

        switch( appState )
        {
         case ZIP_STATE:
            {
                zipSpelling[ 0 ] = '\0';
                zipPhrase[ 0 ] = '\0';
                digitSpelling[ 0 ] = '\0';
                stateSpelling[ 0 ] = '\0';

                zipAgreesWithState = 0;
                int wasGoodZipPhrase = 0;
                bool isNewWord = 0;

                SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), "" );
                SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_ID ), "" );
                SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_ID ), "" );

                recog->setUser( digitsUser );
                recog->setVoc( digitsVoc );
                recog->setState( "digit0" );

                if( 0 == recog->contRecog() )
                {
                    recog->resultName( zipPhrase, sizeof( zipPhrase ) );
                    // zipPhrase is "0 1 9 1 3 massachusetts"

                    wasGoodZipPhrase = isGoodZipPhrase();
                    // zipSpelling is "01913 massachusetts"
                    // digitSpelling is "01913"
                    // stateSpelling is "massachusetts"
                }

                // now do recognition against single word commands...
                SD_WORD  newWord=0;

                recog->setUser( cityUser );
                recog->setVoc( cityVoc );

                recog->setState( "hole1" );

                if( wasGoodZipPhrase )
                {
                    isNewWord = (newWord=recog->wordId( zipSpelling )) == 0;

                    if( isNewWord )
                        newWord = recog->buildWord( zipSpelling, zipPhrase );
                    else
                recog->addWord( newWord );
            }

            if( cancelWord == 0 )
            {
                cancelWord = recog->wordId( "canceladdress" );
                badLabelWord = recog->wordId( "badlabel" );
                goToSleepWord = recog->wordId( "gotosleep" );
                wakeUpWord = recog->wordId( "wakeup" );
            }

            if( recog->recog() == 0 )
            {
                for( int i = 0; i < recog->resultCount(); ++i )
                {
                    recog->resultName( buf, sizeof( buf ), i );

                    SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADDSTRING,
=> 0, (DWORD )buf );
                }

                // clean up
                if( newWord )
                    recog->deleteWord( newWord );

                addressConfidence = recog->confidence();

                if( addressConfidence < 6 ||
                    (wasGoodZipPhrase < 5 && recog->resultId() == newWord) )
                {
                    itsBad( hwnd, &zipUtt );

                    break;
                }
                else if( wasGoodZipPhrase == 6 && recog->resultId() == newWo
=> rd )
                {
                    SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ), digi
=> tSpelling );

                    SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_ID ), st
=> ateSpelling );

                    recog->giveAwayUtt( &zipUtt );

                    zipUtt.setPrompt( zipPhrase );

                    zipValue = atol( digitSpelling );

                    StateZip* sz = StateZip::findStateZip( zipValue );

                    cityHypo.removeAll();

                    cityIds.removeAll();

                    sz->zipToStateCities( zipValue, &zipState, &cityHypo );

                    zipAgreesWithState = !strcmp( zipState, stateSpelling );

                    if( zipAgreesWithState && sz->isPure() )
                    {
                        cityUtt.setPrompt( "pureState" );

                        outCommPort();

                        recog->killUtt();
```

WCDACH.CPP 7-22-95 11:33a

```
                        appState = ZIP_STATE;

                        break;
                    }

                    recog->setUser( cityUser );
                    recog->setVoc( cityVoc );
                    recog->setState( stateSpelling );

                    for( i = cityHypo.count(); i--; )
                    {
                        SD_WORD w = recog->wordId( cityHypo[ i ] );
                        if( w )
                            cityIds.add( w );
                    }

                    appState = CITY_STATE;

                    break;
                }

                if( isCancelBadLabelGoToSleep( hwnd ) )
                {
                    break;
                }

                // else itsBad( hwnd, &zipUtt );

                itsBad( hwnd, &zipUtt );

                break;
            }

            // else recog failed!!!

            // clean up
            if( newWord )
                recog->deleteWord( newWord );

            itsBad( hwnd, &zipUtt );
        }
        break;

    case CITY_STATE:
        {
            // we are pointed at the "right" voc and state

            int i = ( 0 == recog->recog() );

            SD_STATE tmpState = recog->newState( "tmpcity" );

            recog->setState( tmpState );

//          DebugOutput( DBF_ERROR, "Just a Test" );

            if( i )
            {
                for( i = 0; i < recog->resultCount(); ++i )
                    recog->addWord( recog->resultId( i ) );
            }

            for( i = cityIds.count(); i--; )
                recog->addWord( cityIds[ i ] );

            if( 0 == recog->recog() )
```

Page 12 of 19

```
            {
                // cleanup first...
                recog->deleteState( tmpState );

                addressConfidence = recog->confidence();

                for( i = 0; i < recog->resultCount(); ++i )
                {
                    recog->resultName( buf, sizeof( buf ), i );

                    SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADDSTRING,
=> 0, (DWORD )buf );
                }

                if( isCancelBadLabelGoToSleep( hwnd ) )
                {
                    break;
                }

                recog->giveAwayUtt( &cityUtt );

                long foundZip = -1;

                static AC< SD_WORD > cityResultIds;

                cityResultIds.removeAll();

                int cnt = cityUtt.resultCount();

                for( i=0; i < cnt; ++i )
                    cityResultIds.add( cityUtt.resultId( i ) );

                for( i=0; i<cnt; ++i )
                {
                    for( int j=cityIds.count(); j--; )
                    {
                        if( cityIds( j ) == cityResultIds( i ) )
                        {
                            // zip matches some city
                            cityUtt.result( i )->getName( buf, sizeof( buf )
=>  );

                            SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_I
=> D ), buf );

                            cityUtt.setPrompt( buf );

                            if( !zipAgreesWithState )
                            {
                                SetWindowText( GetDlgItem( hwnd, REC_STATE_T
=> EXT_ID ), zipState );

                                strcpy( stateSpelling, zipState );
                            }

                            outCommPort();

                            itsGoodAdapt();

                            goto EndOfSwitch;
                        }
                    }
                }
                // if we got here things didn't match up...
                // try rerecognizing zip with the best city's zips...
```

WCDACH.CPP 7-22-95 11:33a

```
                int foundDistance;

                if( rerecogZip( &foundZip, &foundDistance ) )
                {
                    addressConfidence = 100 - foundDistance;

                    if( addressConfidence < 0 )
                        addressConfidence = 0;

                    if( addressConfidence > 0 )
                    {
                        // everything matches... zip state city
                        const char *s = ltoZip( foundZip );

                        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ),
=> s );

                        cityUtt.result()->getName( buf, sizeof( buf ) );

                        SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_ID ),
=>  buf );

                        cityUtt.setPrompt( buf );

                        zipAgreesWithState = !strcmp( StateZip::findStateNam
=> e( foundZip ),
                                                          stateSpelling );
                        strcpy( digitSpelling, s );

                        outCommPort();

                        for( char *ss=buf; (*ss++ = *s++) != '\0'; )
                            *ss++ = ' ';

                        strcpy( ss-1, stateSpelling );

                        zipUtt.setPrompt( buf );

                        itsGoodAdapt();

                        break;
                    }
                }

                // else its Wrong...
                itsBad( hwnd, &cityUtt );

                break;
            }
            // cleanup first...
            recog->deleteState( tmpState );

            // no recognition...
            itsBad( hwnd, &cityUtt );

            break;
        }

    case WAKEUP_STATE:
        {
            recog->setUser( cityUser );
            recog->setVoc( cityVoc );

            recog->setState( (SD_STATE )0 );            // open Recognition

            if( 0 == recog->recog() )
```

Page 13 of 19

```
            {
                SD_WORD wordToCompareWith = recog->resultId();

                if( wordToCompareWith != wakeUpWord )
                {
                    recog->setState( "wakeup" );        // open Recognition

                    recog->addWord( wordToCompareWith );

                    recog->recog();

                    recog->deleteWord( wordToCompareWith );
                }

                if( recog->resultId() == wakeUpWord )
                {
                    if( recog->resultCount() == 1 ||
                        ( recog->resultCount() > 1 &&
                          recog->resultDistance() > 30 ) )
                    {
                        SetWindowText( GetDlgItem( hwnd, REC_ZIP_TEXT_ID ),
  => "Awake" );

                        recog->beep( "awake.wav" );

                        SetWindowText( GetDlgItem( hwnd, REC_STATE_TEXT_ID )
  => , "" );

                        SetWindowText( GetDlgItem( hwnd, REC_CITY_TEXT_ID ),
  => "" );

                        appState = ZIP_STATE;
                    }
                }
            }

            recog->killUtt();
        }
        break;
    }
EndOfSwitch:
        SetWindowText( GetDlgItem( hwnd, CONFIDENCE_TEXT_ID ), itoa( addressConf
  => idence, buf, 10 ) );

        SetWindowText( GetDlgItem( hwnd, PROMPT_TEXT_ID ), sayPrompt( appState )
  => );

        if( recog->peekUtt() )
            postSpeechEvent( (SD_CHANNEL )1, SD_CHANNEL_START );
    }
}


void DlgWin::onSysCommand( HWND hwnd, UINT cmd, int, int )
{
    switch( cmd )
    {
     case SC_CLOSE:
        EndDialog( hwnd, 0 );
        break;
    }
}


bool findOrCreate( char* destFile, char* srcFile )
{
```

```
    int wfd = open( destFile, O_RDONLY | O_BINARY );

    if( wfd == -1 )
    {
        int rfd = open( srcFile, O_RDONLY | O_BINARY );

        if( rfd == -1 )
            return FALSE;

        wfd = creatnew( destFile, FA_NORMAL );
        close( wfd );

        wfd = open( destFile, O_WRONLY | O_BINARY );

        if( wfd == -1 )
        {
            close( rfd );
            return FALSE;
        }

        int bytesRead;

        char buf[ 2 * 1024 ];

        while( (bytesRead = read( rfd, buf, sizeof(buf) ) ) != 0 )
        {
            while( bytesRead )
                bytesRead -= write( wfd, buf, bytesRead );
        }
        close( rfd );
    }

    close( wfd );

    return TRUE;
}

void DlgWin::setDigitsUser( char* usrName )
{
    strcpy( digitsUser, usrName );

    int len = strlen( digitsUser );

    digitsUser[ len - 2 ] = 'd';

    if( findOrCreate( digitsUser, usrName ) == 0 )
    {
        MessageBox( hWnd, "Could not create a new digits user file",
                  "User Models Error", MB_TASKMODAL | MB_ICONSTOP | MB_OK );

        EndDialog( hWnd, 0 );
    }

    if( tolower( digitsUser[ len -1 ] ) == 'm' )
    {
        cityDigitsVoc = malCityDigitsVoc;
        digitsVoc = malDigitsVoc;
    }
    else
    {
        cityDigitsVoc = femCityDigitsVoc;
        digitsVoc = femDigitsVoc;
    }

    if( findOrCreate( cityDigitsVoc, digitsVoc ) == 0 )
    {
        MessageBox( hWnd, "Could not create a new digits Voc file",
                    "User Vocabulary Error", MB_TASKMODAL | MB_ICONSTOP | MB_OK
  => );

        EndDialog( hWnd, 0 );
    }
}


void DlgWin::onCommand( HWND hwnd, UINT cmd, HWND, UINT )
{
    switch ( cmd )
    {
     case CHOICE_LIST_ID:
        break;

     case USR_COMBO_ID:
        break;

     case SAVE_BTN_ID:
        if( recog && channel && recog->getUser() )
        {
            channel->flush();
            recog->saveUser();
        }
        break;

     case EXIT_BTN_ID:
        EndDialog( hwnd, 0 );
        break;

     case FIX_ERRORS_BTN_ID:

        fixError = !fixError;

        SetWindowText( GetDlgItem( hwnd, FIX_ERRORS_BTN_ID ),
                       fixError ? "Fix &Errors" : "Ignore &Errors" );
        break;



     case SETUSER_BTN_ID:
        {
            char usrName[ UTT_PROMPT_LENGTH ];

            if( !GetDlgItemText( hwnd, USR_COMBO_ID, usrName, sizeof(usrName) )
  => )
                break;

            HCURSOR hCursor = SetCursor( LoadCursor( 0, IDC_WAIT ) );

            bool isMale = tolower( usrName[ strlen( usrName ) - 1 ] ) == 'm';

            if( recog )
            {
                char buffer[ 128 ];
                recog->getUsername( buffer, sizeof( buffer ) );

                if( fileNameCmp( usrName, buffer ) )
                {
                    bool wasListening = channel->isListening();

                    if( wasListening )
                        channel->listen( FALSE );
```

UCOACH.CPP 7-22-95 11:33a

```
                    recog->closeVoc( cityVoc );
                    recog->closeVoc( digitsVoc );
                    recog->closeVoc( cityDigitsVoc );
                    recog->closeUser( cityUser );
                    recog->closeUser( digitsUser );

                    strcpy( cityUser, usrName );

                    setDigitsUser( usrName );

                    if( isMale )
                    {
                        digitsVoc = malDigitsVoc;
                        cityVoc = malCityVoc;
                    }
                    else
                    {
                        digitsVoc = femDigitsVoc;
                        cityVoc = femCityVoc;
                    }

                    if( recog->setUser( cityUser ) == 0 ||
                        recog->setVoc( cityVoc ) == 0 ||
                        recog->setVoc( cityDigitsVoc ) == 0 ||
                        recog->setUser( digitsUser ) == 0 ||
                        recog->setVoc( digitsVoc ) == 0 )
                    {
                        MessageBox( hwnd, "Could not open either digits.voc or c
=> tall.voc or user file!",
                                    DlgWin::name, MB_ICONEXCLAMATION | MB_OK );

                        EndDialog( hwnd, 0 );
                    }

                    if( wasListening )
                        channel->listen( TRUE );
                }
            }
            else        // recog == 0
            {
                if( isMale )
                {
                    digitsVoc = malDigitsVoc;
                    cityVoc = malCityVoc;
                }
                else
                {
                    digitsVoc = femDigitsVoc;
                    cityVoc = femCityVoc;
                }

                if( channel == 0 )
                    channel = new WindowsLiveChannel();

                recog = new Recognizer( channel );

                strcpy( cityUser, usrName );

                setDigitsUser( usrName );

                if( recog->setUser( cityUser ) == 0 ||
                    recog->setVoc( cityVoc ) == 0 ||
                    recog->setVoc( cityDigitsVoc ) == 0 ||
                    recog->setUser( digitsUser ) == 0 ||
                    recog->setVoc( digitsVoc ) == 0 )
```

Page 15 of 19

```
                {
                    MessageBox( hwnd, "Could not open either digits.voc or ctall
=> .voc or user file!",
                                DlgWin::name, MB_ICONEXCLAMATION | MB_OK );

                    EndDialog( hwnd, 0 );
                }

                channel->open( postSpeechEvent );

                EnableWindow( GetDlgItem( hwnd, MICROPHONE_BTN_ID ), TRUE );

//              EnableWindow( GetDlgItem( hwnd, SETUSER_BTN_ID ), FALSE );

                SetFocus( GetDlgItem( hwnd, WORD_PAUSE_SLIDE_ID ) );

                numErrors = 0;
            }

            SetCursor( hCursor );
        }
        break;

    case MICROPHONE_BTN_ID:
        if( recog )
        {
            recog->listen( !recog->isListening() );

            SetWindowText( GetDlgItem( hwnd, MICROPHONE_BTN_ID ),
                           recog->isListening() ? "&Microphone On" : "&Microphon
=> e Off" );

            SetWindowText( GetDlgItem( hwnd, PROMPT_TEXT_ID ),
                           recog->isListening() ? sayPrompt( appState ) : "" );

            if( !recog->isListening() )
            {
                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_RESETCONTENT, 0, 0
=> );

                long total = right + wrong;
                if( total == 0 ) total = 1;

                sprintf( buf, "right:    %ld  %ld\%", right, (100*right)/total )
=> ;
                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADDSTRING, 0, (DWOR
=> D )buf );
                sprintf( buf, "wrong:    %ld  %ld\%", wrong, (100*wrong)/total )
=> ;
                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADDSTRING, 0, (DWOR
=> D )buf );
                sprintf( buf, "predicted: %ld  %ld\%", predicted, (100*predicted
=> )/total );
                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADDSTRING, 0, (DWOR
=> D )buf );

                KillTimer( hwnd, VUTIMER_ID );
            }
            else
            {
                // initialize vu meter
                SetTimer( hwnd, VUTIMER_ID, 20, (TIMERPROC )0 );
            }

        }
        break;
```

WDOACN.CPP 7-22-95 11:33a

```
    case WORD_PAUSE_SLIDE_ID:
        {
            wordPause = GetScrollPos( GetDlgItem( hwnd, WORD_PAUSE_SLIDE_ID ), S
=> B_CTL );

            if( recog )
            {
                bool wasListening = recog->isListening();

                if( wasListening )
                    recog->listen( 0 );

                speechTask->setPar( "word-pause", (short )wordPause );

                if( wasListening )
                    recog->listen( 1 );
            }
        }
        break;
    }
}


void DlgWin::onTimerDrawVU( HWND hwnd, UINT id )
{
    if( id == VUTIMER_ID && channel->isListening() )
    {
        bool test;

        FEP_S_STATUS fepinfo;

        Fep_Status( (BOOL* )&test, &fepinfo ) ;

        HWND hvu = GetDlgItem( hwnd, VUMETER_ID );

        RECT rc;

        GetClientRect( hvu, &rc );

//      rc.left     = 95;
//      rc.right    = 95 + 61;
//      rc.top      = 116;
//      rc.bottom   = 116 + 3;

        HDC hdc = GetDC( hvu );

        int noiseright, speechleft, speechright, unit;

        unit = (rc.right - rc.left) / fepinfo.max_level;

        noiseright = speechleft = unit * fepinfo.noise_level;

        speechright = unit * fepinfo.speech_level + speechleft;

        //      draw speech
        HBRUSH hBrush = CreateSolidBrush( RGB( 255, 0, 0 ) );

        HBRUSH hOldBrush = (HBRUSH )SelectObject( hdc, hBrush );

        Rectangle( hdc, rc.left, rc.top, noiseright, rc.bottom );

        SelectObject( hdc, hOldBrush );

        DeleteObject(hBrush);
```

Page 16 of 19

```
        //      draw speech
        hBrush = CreateSolidBrush( RGB( 0, 255, 0 ) );

        hOldBrush = (HBRUSH )SelectObject( hdc, hBrush );

        Rectangle( hdc, speechleft, rc.top, speechright, rc.bottom );

        SelectObject( hdc, hOldBrush );

        DeleteObject( hBrush );

        // Background
        Rectangle( hdc, speechright, rc.top , rc.right, rc.bottom );

        ReleaseDC( hvu, hdc );
    }
}


void DlgWin::onDestroy( HWND hwnd )
{
    KillTimer( hwnd, VUTIMER_ID );

    if( speechTask )
    {
        zipUtt.killUtt();
        cityUtt.killUtt();

        if( channel )
        {
            delete channel;
            channel = 0;
        }

        if( recog )
        {
            delete recog;
            recog = 0;

            choiceListStats.printStats();
        }

        delete speechTask;
        speechTask = 0;
    }

    for( int i=0; i<4; ++i )
    {
        if( comm[ i ] )
        {
            delete comm[ i ];

            comm[ i ] = 0;
        }
        if( commFormat[ i ] )
        {
            delete commFormat[ i ];

            commFormat[ i ] = 0;
        }
    }

//    PostQuitMessage( 0 );
}
```

WCDACH.CPP 7-22-95 11:33a

```
long FAR PASCAL _export DlgWin::wndProc( HWND hwnd, UINT message, WPARAM wParam,
  => LPARAM lParam )
{
    messageHandled = TRUE;

    long returnValue = 0;

    switch( message )
    {
        handleMessage( WM_INITDIALOG, hwnd, wParam, lParam, onInitDialog );
        handleMessage( WM_DESTROY, hwnd, wParam, lParam, onDestroy );
        handleMessage( WM_COMMAND, hwnd, wParam, lParam, onCommand );
        handleMessage( WM_SYSCOMMAND, hwnd, wParam, lParam, onSysCommand );
        handleMessage( WM_TIMER, hwnd, wParam, lParam, onTimerDrawVU );
        handleMessage( UM_CHANNELSTART, hwnd, wParam, lParam, onSpeech );
        handleMessage( UM_COMM_CHECK_MSG, hwnd, wParam, lParam, onCommCheckMsg )
  => ;
        handleMessage( UM_COMM_SEND_SYNC_MSG, hwnd, wParam, lParam, onCommSendSy
  => ncMsg );
    }

    return messageHandled ? returnValue : 0;
}



////////////////////////////////////////////////////////////////////////
// AppWin
////////////////////////////////////////////////////////////////////////

class AppWin
{
 public:
    static long FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );

    static HWND  hWnd;
    static HTASK hTask;

 protected:

    static bool messageHandled;

    // wndProc() services
    static bool onCreate( HWND hWnd, CREATESTRUCT FAR* lpCreateStruct );
    static void onDestroy( HWND hWnd );
    static void onNCDestroy( HWND hWnd );
    static void onClose( HWND hWnd );
    static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y );


    friend int PASCAL WinMain( HINSTANCE, HINSTANCE, LPSTR, int );
};


// Static Global Variables of AppWin
HWND            AppWin::hWnd;
HTASK           AppWin::hTask;
bool            AppWin::messageHandled;



int PASCAL WinMain( HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR lpszCmdLine,
```

Page 17 of 19

```
=>  int /* nCmdShow */ )
{
    static const char name[] = "DragCpp";

    SetMessageQueue( 64 );

    if( !hPrevInstance )
    {
        WNDCLASS wndclass;

        wndclass.style          = CS_HREDRAW | CS_VREDRAW;
        wndclass.lpfnWndProc    = AppWin::wndProc;
        wndclass.cbClsExtra     = 0;
        wndclass.cbWndExtra     = 0;
        wndclass.hInstance      = hInst;
        wndclass.hIcon          = LoadIcon( 0, MAKEINTRESOURCE( IDI_MY_ICON ) );
        wndclass.hCursor        = LoadCursor( 0, IDC_ARROW );
        // wndclass.hIcon          = LoadIcon( hInst, "merc.ico" );
        // wndclass.hCursor        = LoadCursor( 0, IDC_ARROW );
        wndclass.hbrBackground = 0;
        wndclass.lpszMenuName  = 0;
        wndclass.lpszClassName = name;

        RegisterClass( &wndclass );
    }

    ::hInstance = hInst;        // the global hInstance for the sliderctl
    slideControl.libMainInit(); // register the slider class

    HWND hWnd = CreateWindow( name,
                              name,
                              WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
                              CW_USEDEFAULT,
                              CW_USEDEFAULT,
                              CW_USEDEFAULT,
                              CW_USEDEFAULT,
                              0,
                              0,
                              hInst,
                              lpszCmdLine );

    assert( hWnd == AppWin::hWnd );

    ShowWindow( AppWin::hWnd, SW_HIDE );
//    ShowWindow( hWnd, nCmdShow );
//    UpdateWindow( hWnd );

    MSG msg;

    while( GetMessage( &msg, 0, 0, 0 ) )
    {
        TranslateMessage( &msg );
        DispatchMessage( &msg );
    }

    return msg.wParam;
}



void AppWin::onCommand( HWND hWnd, UINT cmd, HWND, UINT )
{
    switch ( cmd )
    {
     case UM_INIT_DIALOG:
```

MCOACH.CPP 7-22-95 11:33a

```
        {
            // create dialog box proc instances for use later
            static FARPROC lpfn = MakeProcInstance( (FARPROC )DlgWin::wndProc, h
  => Instance );

            assert( lpfn );

            DialogBox( hInstance, "TestDialog", 0, (DLGPROC )lpfn );

            PostMessage( hWnd, WM_CLOSE, 0, 0 );
        }
        break ;


      default:
        messageHandled = FALSE;
    }
}


bool AppWin::onCreate( HWND hwnd, CREATESTRUCT FAR* )
{
    hTask = GetCurrentTask();

    hWnd = hwnd;

    PostMessage( hwnd, WM_COMMAND, UM_INIT_DIALOG, 0 );

    return TRUE;
}


void AppWin::onDestroy( HWND )
{
    PostQuitMessage( 0 );
}

void AppWin::onNCDestroy( HWND )
{
}

void AppWin::onClose( HWND hwnd )
{
    DestroyWindow( hwnd );
}


long FAR PASCAL _export AppWin::wndProc( HWND hwnd, UINT message, WPARAM wParam,
  =>  LPARAM lParam )
{
    messageHandled = TRUE;

    long returnValue = 0;

    switch( message )
    {
        handleMessage( WM_CREATE, hwnd, wParam, lParam, onCreate );
        handleMessage( WM_DESTROY, hwnd, wParam, lParam, onDestroy );
        handleMessage( WM_NCDESTROY, hwnd, wParam, lParam, onNCDestroy );
        handleMessage( WM_CLOSE, hwnd, wParam, lParam, onClose );
        handleMessage( WM_COMMAND, hwnd, wParam, lParam, onCommand );

     'Default:
        messageHandled = FALSE;
```

Page 18 of 19

```
            break;
        }

    return( messageHandled
            ? returnValue
            : DefWindowProc( hwnd, message, wParam, lParam ) );
}
```

```
/*****************************************************************************
        Description

        arrcoll.cpp
        base classes to manage Array Collection

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995

*****************************************************************************/

#include "arrcoll.h"
#include "defs.h"


#if DEBUG || UNIT_1


ACBase::ACBase( size_t entSize, unsigned iniSize, unsigned incSize ) {

    assert( entSize );

    entrySize = entSize;

    arraySize = ( iniSize <= 0 ) ?  6  :  iniSize;

    incrementSize = ( incSize <= 0 ) ?  arraySize  :  incSize;

    array = new char[ arraySize * entrySize ];

    arrayUsed = 0;
}


#endif
#if DEBUG || UNIT_2


int ACBase::add( const char& e, int index )
{
    index = makeRoom( 1, index );

    if( index == -1 )
        return -1;

    memcpy( array +index*entrySize, &e, entrySize );

    return index;
}


#endif
#if DEBUG || UNIT_3


void ACBase::removeEntry( unsigned index )
{
    if( index < arrayUsed )
    {
        memmove( array + index*entrySize, array + (index + 1)*entrySize,
                 (arrayUsed - index -1) * entrySize );

        --arrayUsed;
    }
```

ARRCOLL.CPP 7-22-95 11:35a

```
}


#endif
#if DEBUG || UNIT_5


int ACBase::find( const char& v ) const
{
    for( int i=0, ii=0; i < arrayUsed; ++i, ii+=entrySize )
    {
        if( !memcmp( (void*) &v, array + ii, entrySize ) ) return i;
    }

    return -1;
}


#endif
#if DEBUG || UNIT_6


int ACBase::find( const char& v, CompareACEntries cmp,
                                 int sortedList ) const
{
    assert( cmp );


    if( sortedList )
        return find( v, cmp, &sortedList );
    else
    {
        for( int i=0, ii=0; i < arrayUsed; ++i, ii+=entrySize )
        {
            if( !cmp( (void*) &v, array + ii ) ) return i;
        }
    }

    return -1;
}


#endif
#if DEBUG || UNIT_7

int ACBase::find( const char& e, CompareACEntries compare,
                                 int* position ) const
{
    // returns -1 if it cannot find the entry
    // if the compare function is given, it does a log2 search
    // if position is given and the entry is not found, position will
    // be the proper position this item should be added at;

    assert( compare );

    int testPosition=0, lowIndex=0, highIndex=count() -1, cmp=-1;

    static int lastFound = -1;

    if( lastFound > highIndex || lastFound < 0 )
        lastFound = ( lowIndex + highIndex + 1 ) / 2;

    while( lowIndex <= highIndex )
    {
        testPosition = lastFound;
```

Page 1 of 2

```
        cmp = compare( (void*) &e, (void*) (array + (testPosition*entrySize)) );

        if( cmp == 0 ) break;

        else if( cmp > 0 )
            lowIndex = testPosition + 1;

        else    // cmp < 0
            highIndex = testPosition - 1;

        lastFound = ( lowIndex + highIndex ) / 2;
    }

    if( position ) *position = testPosition + (cmp > 0);

    return (cmp == 0) ? testPosition : -1 ;
}

#endif
#if DEBUG || UNIT_8

int ACBase::indexOf( const char& e ) const
{
    if( &e>=array && &e<=lastEntry() ) return (&e - array)/entrySize;

    return -1;
}

#endif
#if DEBUG || UNIT_9

int ACBase::makeRoom( int numToAdd, int index )
{
    assert( arrayUsed <= arraySize );

    assert( numToAdd > 0 );

    if( index == -1 )
        index = arrayUsed;

    assert( index <= arrayUsed ); //  if( index > arrayUsed ) index=arrayUsed;

    if( arrayUsed+numToAdd > arraySize )
    {
        unsigned newSize = arraySize + incrementSize;

        if( newSize < arraySize+numToAdd )
            newSize=arraySize+numToAdd;

        if( newSize > ((unsigned)0xffff / entrySize) ) {
            assert( newSize < ((unsigned)0xffff / entrySize) );
            return -1;
        }

        char *newArray = new char[ newSize * entrySize ];

        if( !newArray ) {
            assert( newArray );
            return -1;
        }

        incrementSize += (incrementSize>>2) + 48/(1 + (47/incrementSize) );
```

ARRCOLL.CPP 7-22-95 11:35a

```
        memcpy( newArray, array, index*entrySize );

        memcpy( newArray +(index + numToAdd)*entrySize,
                array + index*entrySize,
                (arrayUsed - index)*entrySize );

        delete [] array;

        array = newArray;

        arraySize = newSize;
    }
    else
    {
        memmove( array +(index+numToAdd)*entrySize,
                array + index*entrySize,
                (arrayUsed - index)*entrySize );
    }

    arrayUsed += numToAdd;

    return index;
}

#endif
```

Page 2 of 2

```
////////////////////////////////////////////////////////////////////////
/*

    Description

        WTRAIN.CPP

        Training app to train USR files to user's voice. Not done in noise.

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995

***********************************************************************/

#include <windowsx.h>
#include <string.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <ctype.h>
#include "dragcpp.h"

#include "wtrain.h"

#include "slidectl.h"

extern "C" (
#include "fepext.h"
)

// this is the instance of the control
DgnSlideControl slideControl;
extern HINSTANCE hinstance;        // when we call DgnSlideControl::libMainInit()
HINSTANCE hinstance;               // hinstance must be initialized...


,static char femCityVoc[]          = "ctall.vof";
 static char femDigitsVoc[]        = "digits.vof";
 static char malCityVoc[]          = "ctall.vom";
 static char malDigitsVoc[]        = "digits.vom";
 static char* cityVoc = 0;
 static char* digitsVoc = 0;



#define LINE_SIZE 80

#define handleMessage( message, hWnd, wParam, lParam, fn ) \
    case (message) : returnValue = HANDLE_ ## message( (hWnd), (wParam), (lParam
  => ), (fn) ); break;



class DlgWin
(
 public:
    static long FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );
    static HWND  hWnd;

 protected:

    static bool messageHandled;
    static bool sawError;
    static bool ignoreErrors;
```

WTRAIN.CPP 7-22-95 11:35a

```
    static char* name;
    static short wordPause;

    static SpeechTask *speechTask;                        // speech driver task ID
    static UttChannel *channel;
    static Recognizer* recog;

    static void reportError( int code, char far *message );
    static void SD_CALLBACK _export postSpeechEvent( SD_CHANNEL, SD_CHANNEL_EVEN
  => T );
    static void setupPrompt( HWND hWnd );

    // wndProc() services
    static bool onInitDialog( HWND hWnd, HWND topCntrl, long );
    static void onDestroy( HWND hWnd );
    static void onClose( HWND hWnd ;;
    static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y );
    static void onSpeech( HWND hWnd, SD_CHANNEL ch );
    static void onTimerDrawVU( HWND hwnd, UINT id );

    friend int PASCAL WinMain( HINSTANCE, HINSTANCE, LPSTR, int );
);


HWND            DlgWin::hWnd;
bool            DlgWin::messageHandled;
bool            DlgWin::sawError;
bool            DlgWin::ignoreErrors;
char*           DlgWin::name = "dragon123\0      ";
short           DlgWin::wordPause=256;
SpeechTask*     DlgWin::speechTask;
Recognizer*     DlgWin::recog;
UttChannel*     DlgWin::channel;



////////////////////////////////////////////////////////////////////////
// configuration file stuff
////////////////////////////////////////////////////////////////////////


struct ConfigCommands
(
    const char* name;
    char* (*command)( char* );
);


static char* setCommand( char* );

static ConfigCommands cfgCommands[] = (
    ( "set", setCommand ),
    ( 0, 0 }
);


static char* setNothing( char* )
(
    return 0;
)


static char userPathString[ 256 ] = ".";
static char* setUserPath( char* value )
(
```

Page 1 of 8

```
    for( char *s = value; *s && !isspace( *s ); ++s );

    *s = '\0';

    strcpy( userPathString, value );

    return 0;
}

static char* setFepPar( WORD id, char* value )
{
    for( char *s = value; *s && !isspace( *s ); ++s );

    *s = '\0';

    if( *value )
    {
        WORD v;

        v = (WORD )atol( value );

        fep_Setpar( SUPER_USER, id, v );

        return 0;
    }
    return "Par Not Set";
}

static char* setStartOfSpeech( char* value )
{
    if( setfepPar( 13, value ) )
        return "StartOfSpeech Not Set";

    return 0;
}

static char* setEndOfSpeech( char* value )
{
    if( setfepPar( 12, value ) )
        return "EndOfSpeech Not Set";

    return 0;
}

static ConfigCommands setCommands[] = {
    { "Com",                    setNothing },
    { "Comm",                   setNothing },
    { "ComPort",                setNothing },
    { "Pure",                   setNothing },
    { "UserPath",               setUserPath },
    { "StartOfSpeech",          setStartOfSpeech },
    { "EndOfSpeech",            setEndOfSpeech },

    { "ZipRange",               setNothing },

    { "BadLabel",               setNothing },
    { "FirstDayAir",            setNothing },
    { "NextDayAir",             setNothing },

    { "Alberta",                setNothing },
    { "BritishColumbia",        setNothing },
    { "Labrador",               setNothing },
    { "Manitoba",               setNothing },
    { "NewBrunswick",           setNothing },
    { "Newfoundland",           setNothing },
    { "NorthWestTerritories",   setNothing },
```

```
    { "NovaScotia",             setNothing },
    { "Ontario",                setNothing },
    { "PrinceEdwardIsland",     setNothing },
    { "Quebec",                 setNothing },
    { "Saskatchewan",           setNothing },
    { "YukonTerritory",         setNothing },

    { "Mexico",                 setNothing },

    { 0, 0 }
};


static char* setCommand( char* var )
{
    while( isspace( *var ) )
        ++var;

    char* value = var;

    while( isalnum( *value ) || *value == '_' )
        ++value;

    if( *value == '\0' )
        return "no value found in set command... ex: set variable = value";

    bool foundEqual = (*value == '=');

    *value++ = '\0';

    if( !foundEqual )
    {
        while( isspace( *value ) )
            ++value;

        if( *value != '=' )
            return "no \"=\" sign in set command... ex: set variable = value";

        ++value;
    }

    while( *value && isspace( *value ) )
        ++value;

    for( int i=0; setCommands[ i ].name != 0; ++i )
    {
        if( stricmp( var, setCommands[ i ].name ) == 0 )
            return setCommands[ i ].command( value );
    }

    return "unknown set variable";
}


void parseConfig( const char* filename )
{
    FILE *cfgFile = fopen( filename, "r" );

    char *errorMessage = ( cfgFile == 0
                           ? "could not open UPS.CFG file"
                           : 0 );
    char line[ 256 ];
    int lineNumber = 0;

    while( errorMessage == 0 && fgets( line, sizeof( line ), cfgFile ) != 0 )
    {
        char *var = line;

        ++lineNumber;

        while( isspace( *var ) )
            ++var;

        if( *var == '#' || *var == '\0' )
            continue;

        char *value = var;

        while( !isspace( *value ) && *value != '\0' )
            ++value;

        if( *value != '\0' )
        {
            *value++ = '\0';

            while( isspace( *value ) )
                ++value;
        }

        for( int i=0; cfgCommands[ i ].name != 0; ++i )
        {
            if( stricmp( var, cfgCommands[ i ].name ) == 0 )
            {
                errorMessage = cfgCommands[ i ].command( value );

                break;
            }
        }
    }

    if( errorMessage )
    {
        char errorType[ 40 ] = "UPS.CFG error, line           \n";

        itoa( lineNumber, errorType + 20, 10 );

        MessageBox( DlgWin::hWnd, errorMessage, errorType, MB_OK | MB_ICONEXCLAM
=> ATION );
    }

    if( cfgFile )
        fclose( cfgFile );
}



bool DlgWin::onInitDialog( HWND hDlg, HWND, long )
{
    hWnd = hDlg;

    speechTask = new SpeechTask( name, reportError );

    recog = 0;

    struct ffblk attrib;

    // set up the usr listbox
    if( !findfirst( "*.un?", &attrib, FA_NORMAL ) )
    {
        do {
```

```
            SendDlgItemMessage( hDlg, USR_COMBO_ID, CB_ADDSTRING, 0, (DWORD) att
=> rib.ff_name );
        } while( !findnext( &attrib ) );
    }

    // set up the prompt listbox
    if( !findfirst( "*.txt", &attrib, FA_NORMAL ) )
    {
        do {
            SendDlgItemMessage( hDlg, PROMPT_COMBO_ID, CB_ADDSTRING, 0, (DWORD)
=> attrib.ff_name );
        } while( !findnext( &attrib ) );
    }

    speechTask->setPar( "adapt-tolerance", (short )3 );

    SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLRANGE, 0, 0x02000
=> 064L );

    SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLPOS, 0x0100, 1 );

    return TRUE;
}



void DlgWin::reportError( int code, char far *message )
{
    sawError = YES;
    if( ignoreErrors )
        return;          // prevent recursion
    ignoreErrors = YES;

    // Throw up an error message
    char buffer[ 512 ];
    sprintf( buffer, "An error has occurred\n  code = %d\n  message = %s",
             code, message );

    if( stricmp( message, "Bad token" ) )
    {
        MessageBox( hWnd, buffer, "Wintalk SDAPI Error",
                    MB_TASKMODAL | MB_ICONSTOP | MB_OK );

        // terminate...  kill the hTopParentWindow
        if( hWnd )
            PostMessage( hWnd, WM_CLOSE, 0, 0 );
    }
    ignoreErrors = 0;
}

void SD_CALLBACK _export DlgWin::postSpeechEvent( SD_CHANNEL ch, SD_CHANNEL_EVEN
=> T eventIn )
{
    assert( eventIn == SD_CHANNEL_START );

    MSG msg;

    if( 0 == PeekMessage( &msg, hWnd, UM_CHANNELSTART, UM_CHANNELSTART, PM_NOREM
=> OVE | PM_NOYIELD ) )
        PostMessage( hWnd, UM_CHANNELSTART, (WORD )ch, 0 );

} // EventHandler

#define HANDLE_UM_CHANNELSTART(hwnd, wParam, lParam, fn) \
                                   ((fn)(hwnd, (SD_CHANNEL )wParam), 0L)
```

```
int numDigits( const char* s )
{
    for( int len = 0; *s; ++s )
    {
        if( isdigit( *s ) )
            ++len;

        else if( !isspace( *s ) )
            return 0;
    }
    return len;
}

void DigWin::setupPrompt( HWND hWnd )
{
    char zipBufP[ LINE_SIZE ];
    char stateBufP[ LINE_SIZE ];
    char cityBufP[ LINE_SIZE ];

    int i = recog->promptIndex() % 3;        // -1 == EOF

    int promptOffsets[] = { -2, -1, 0, 1, 2 };

    int* peekOffset = promptOffsets + 2 - i;

    if( i != -1 && recog->peekPrompt( cityBufP, LINE_SIZE, peekOffset[ 2 ] ) )
    {
        recog->peekPrompt( stateBufP, LINE_SIZE, peekOffset[ 1 ] );

        recog->peekPrompt( zipBufP, LINE_SIZE, peekOffset[ 0 ] );
    }
    else
    {
        zipBufP[ 0 ] = '\0';
        stateBufP[ 0 ] = '\0';
        cityBufP[ 0 ] = '\0';

        if( i == -1 && recog->isListening() )
            PostMessage( hWnd, WM_COMMAND, MICROPHONE_BTN_ID, 0 );

        else i = -1;
    }

    if( !recog->isListening() )
        i = -1;



    SetWindowText( GetDlgItem( hWnd, PROMPT_ZIP_TXT_ID ), zipBufP );

    SetWindowText( GetDlgItem( hWnd, PROMPT_STATE_TXT_ID ), stateBufP );

    SetWindowText( GetDlgItem( hWnd, PROMPT_CITY_TXT_ID ), cityBufP );


    SetWindowText( GetDlgItem( hWnd, SAY_ZIP_TXT_ID ),
                   ( i == 0 ? "Say:" : "" ) );
    SetWindowText( GetDlgItem( hWnd, SAY_STATE_TXT_ID ),
                   ( i == 1 ? "Say:" : "" ) );
    SetWindowText( GetDlgItem( hWnd, SAY_CITY_TXT_ID ),
                   ( i == 2 ? "Say:" : "" ) );
}

void DigWin::onSpeech( HWND hWnd, SD_CHANNEL ch )
{
```

```
    // get the utterance handle; do not wait for the end of the
    // utterance since the recognition can proceed in parallel
    // with the utterance collection

    assert( recog );

    if( recog->notifyChannel( ch, SD_CHANNEL_START ) )
    {
        char buf[ LINE_SIZE ];

        SendDlgItemMessage( hWnd, CHOICE_LIST_ID, LB_RESETCONTENT, 0, 0 );

        const char* s = recog->prompt();

        int digitLen = numDigits( s );

        if( digitLen )
        {
            recog->setVoc( digitsVoc );
            recog->setState( "digits" );
            recog->seperateDigits( digitLen );
        }
        else
        {
            recog->setVoc( cityVoc );
            recog->setState( (SD_STATE )0 );
        }

        if( 0 == ( digitLen ? recog->contRecog(digitLen) : recog->recog() ) )
        {
            char buf[ LINE_SIZE ], *b;

            int correct = -1;

            for( int i = 0; i < recog->resultCount(); ++i )
            {
                recog->resultName( buf, sizeof( buf ), i );

                SendDlgItemMessage( hWnd, CHOICE_LIST_ID, LB_ADDSTRING, 0, (DWOR
=> D )buf );

                if( !strcmp( buf, recog->prompt() ) )
                    correct = i;
            }

            if( correct == -1 )
                recog->beep( "bad.wav", SND_TOGGLEMIC );

            itoa( recog->confidence(), buf, 10 );

            SetWindowText( GetDlgItem( hWnd, CONFIDENCE_TEXT_ID ), buf );

            recog->nextUtt();

            recog->nextPrompt();

            setupPrompt( hWnd );

            static bool backupDisabled = TRUE;

            if( backupDisabled )
            {
                EnableWindow( GetDlgItem( hWnd, BACKUP_BTN_ID ), TRUE );
                backupDisabled = 0;
            }
            goto On_Speech_Exit;
        }

        recog->beep( "ding.wav" );

        recog->killUtt();
    }

On_Speech_Exit:

    if( recog->peekUtt() )
        postSpeechEvent( ch, SD_CHANNEL_START );

}


void DlgWin::onDestroy( HWND )
{
    if( speechTask )
    {
        if( channel )
        {
            delete channel;
            channel = 0;
        }

        if( recog )
        {
            delete recog;
            recog = 0;
        }

        delete speechTask;
        speechTask = 0;
    }

    PostQuitMessage( 0 );
}

void DlgWin::onClose( HWND hWnd )
{
    DestroyWindow( hWnd );
}



long FAR PASCAL _export DlgWin::wndProc( HWND hWnd, UINT message, WPARAM wParam,
  <> LPARAM lParam )
{
    messageHandled = TRUE;

    long returnValue = 0;

    switch( message )
    {
        handleMessage( WM_INITDIALOG, hWnd, wParam, lParam, onInitDialog );
        handleMessage( WM_DESTROY, hWnd, wParam, lParam, onDestroy );
        handleMessage( WM_CLOSE, hWnd, wParam, lParam, onClose );
        handleMessage( WM_COMMAND, hWnd, wParam, lParam, onCommand );
        handleMessage( UM_CHANNELSTART, hWnd, wParam, lParam, onSpeech );
        handleMessage( WM_TIMER, hWnd, wParam, lParam, onTimerDrawVU );

     default:
        messageHandled = FALSE;

        break;
    }
```

VTRAIN.CPP 7-22-95 11:35a

```
    return messageHandled ? returnValue : 0;
}


/////////////////////////////////////////////////////////////////////
// AppWin
/////////////////////////////////////////////////////////////////////

class AppWin
{
 public:
    static long FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );

    static HWND  hWnd;
    static HTASK hTask;

 protected:

    static bool messageHandled;

    // wndProc() services
    static bool onCreate( HWND hWnd, CREATESTRUCT FAR* lpCreateStruct );
    static void onDestroy( HWND hWnd );
    static void onClose( HWND hWnd );
    static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y );

    friend int PASCAL WinMain( HINSTANCE, HINSTANCE, LPSTR, int );
};


// Static Global Variables of AppWin
HWND            AppWin::hWnd;
HTASK           AppWin::hTask;
bool            AppWin::messageHandled;



static void makeTokName( char* tokName, char* usrName, char* promptName )
{
    char* p = stpcpy( tokName, usrName );

    p -= 4;

    assert( p > tokName );

    for( char* s = promptName + strlen( promptName );
          s > promptName;
          --s )
    {
        if( *s == '\\' || *s == ':' || *s == '/' )
        {
            ++s;
            break;
        }
    }
    while( *s && *s != '.' )
        *p++ = *s++;

    *p = '\0';

    s = p;

    while( s > tokName )
```

Page 5 of 8

```
    {
        if( *s == '\\' || *s == ':' || *s == '/' )
        {
            ++s;
            break;
        }
        --s;
    }
    if( p - s > 7 )
        p = s + 8;

    strcpy( p, ".tok" );
}


void DlgWin::onCommand( HWND hWnd, UINT cmd, HWND, UINT )
{
    switch ( cmd )
    {
     case CHOICE_LIST_ID:
        break;

     case USR_COMBO_ID:
        break;

     case SAVEEXIT_BTN_ID:
        if( recog && channel && recog->getUser() )
        {
            channel->flush();
            recog->saveUser();
        }

     case EXIT_BTN_ID:
        EndDialog( hWnd, 0 );
        break;

     case BACKUP_BTN_ID:
        {
            int maxBack = (recog->promptIndex() % 3);

            if( maxBack == 0 ) maxBack = 3;

            if( recog->getChannel()->canBackup( maxBack ) )
            {
                recog->notifyChannel( 0, SD_CHANNEL_BACKUP );

                recog->nextUtt();

                recog->nextPrompt();

                setupPrompt( hWnd );
            }
        }
        break;

     case MICROPHONE_BTN_ID:
        if( recog == 0 )
        {
            char usrName[ LINE_SIZE ], promptName[ LINE_SIZE ], tokName[ LINE_SI
=> ZE ];

            if( !GetDlgItemText( hWnd, USR_COMBO_ID, usrName, LINE_SIZE ) )
                return;

            if( !GetDlgItemText( hWnd, PROMPT_COMBO_ID, promptName, LINE_SIZE )
```

```
=> )
                return;

            switch( usrName[ strlen( usrName ) - 1 ] )
            {
             case 'm':
             case 'M':
                digitsVoc = malDigitsVoc;
                cityVoc = malCityVoc;
                break;
             case 'f':
             case 'F':
                digitsVoc = femDigitsVoc;
                cityVoc = femCityVoc;
                break;

             default:
                return;
            }

            HCURSOR hCursor = SetCursor( LoadCursor( 0, IDC_WAIT ) );

            makeTokName( tokName, usrName, promptName );

            channel = new WindowsAdaptChannel( promptName );

            recog = new Recognizer( channel );

            recog->setUser( usrName );
            recog->setVoc( digitsVoc );
            recog->setVoc( cityVoc );

            channel->open( postSpeechEvent );

            SetWindowText( GetDlgItem( hWnd, MICROPHONE_BTN_ID ), "Microphone Of
=> f" );

            SetCursor( hCursor );
        }
        else
        {
            recog->listen( !recog->isListening() );

            SetWindowText( GetDlgItem( hWnd, MICROPHONE_BTN_ID ),
                           recog->isListening() ? "Microphone On" : "Microphone
=> Off" );

            if( recog->isListening() )
            {
                if( recog->prompt() == 0 )
                    recog->nextPrompt();

                // initialize vu meter
                SetTimer( hWnd, VUTIMER_ID, 20, (TIMERPROC )0 );
            }
            else
                KillTimer( hWnd, VUTIMER_ID );


            setupPrompt( hWnd );

        }
        break;

     case WORD_PAUSE_SLIDE_ID:
        {
            wordPause = GetScrollPos( GetDlgItem( hWnd, WORD_PAUSE_SLIDE_ID ), S
    => B_CTL );

            if( recog )
            {
                bool wasListening = recog->isListening();

                if( wasListening )
                    recog->listen( 0 );

                speechTask->setPar( "word-pause", (short )wordPause );

                if( wasListening )
                    recog->listen( 1 );
            }
        }
        break;
    }
}


void DlgWin::onTimerDrawVU( HWND hwnd, UINT id )
{
    if( id == VUTIMER_ID && channel->isListening() )
    {
        bool test;

        FEP_S_STATUS fepinfo;

        Fep_Status( (BOOL* )&test, &fepinfo ) ;

        HWND hvu = GetDlgItem( hwnd, VUMETER_ID );

        RECT rc;

        GetClientRect( hvu, &rc );

//      rc.left     = 95;
//      rc.right    = 95 + 61;
//      rc.top      = 116;
//      rc.bottom   = 116 + 3;

        HDC hdc = GetDC( hvu );

        int noiseright, speechleft, speechright, unit;

        unit = (rc.right - rc.left) / fepinfo.max_level;

        noiseright = speechleft = unit * fepinfo.noise_level;

        speechright = unit * fepinfo.speech_level + speechleft;

        //      draw speech
        HBRUSH hBrush = CreateSolidBrush( RGB( 255, 0, 0 ) );

        HBRUSH hOldBrush = (HBRUSH )SelectObject( hdc, hBrush );

        Rectangle( hdc, rc.left, rc.top, noiseright, rc.bottom );

        SelectObject( hdc, hOldBrush );

        DeleteObject(hBrush);

        //      draw speech
        hBrush = CreateSolidBrush( RGB( 0, 255, 0 ) );
```

VTRAIN.CPP 7-22-95 11:35a

```
        hOldBrush = (HBRUSH )SelectObject( hdc, hBrush );

        Rectangle( hdc, speechleft, rc.top, speechright, rc.bottom );

        SelectObject( hdc, hOldBrush );

        DeleteObject( hBrush );

        // Background
        Rectangle( hdc, speechright, rc.top , rc.right, rc.bottom );

        ReleaseDC( hvu, hdc );
    }
}


int PASCAL WinMain( HINSTANCE hinst, HINSTANCE hPrevInstance, LPSTR lpszCmdLine,
  =>  int /* nCmdShow */ )
{
    static const char name[] = "DragCpp";

    SetMessageQueue( 32 );

    if( !hPrevInstance )
    {
        WNDCLASS wndclass;

        wndclass.style          = CS_HREDRAW | CS_VREDRAW;
        wndclass.lpfnWndProc    = AppWin::wndProc;
        wndclass.cbClsExtra     = 0;
        wndclass.cbWndExtra     = 0;
        wndclass.hInstance      = hinst;
        wndclass.hIcon          = LoadIcon( 0, MAKEINTRESOURCE( IDI_MY_ICON ) );
        wndclass.hCursor        = LoadCursor( 0, IDC_ARROW );
        // wndclass.hIcon         = LoadIcon( hinst, "merc.ico" );
        // wndclass.hCursor       = LoadCursor( 0, IDC_ARROW );
        wndclass.hbrBackground = 0;
        wndclass.lpszMenuName  = 0;
        wndclass.lpszClassName = name;

        RegisterClass( &wndclass );
    }

    ::hinstance = hinst;         // the global hinstance for the sliderctl
    slideControl.libMainInit(); // register the slider class

    HWND hWnd = CreateWindow( name,
                              name,
                              WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
                              CW_USEDEFAULT,
                              CW_USEDEFAULT,
                              CW_USEDEFAULT,
                              CW_USEDEFAULT,
                              0,
                              0,
                              hinst,
                              lpszCmdLine );

    assert( hWnd == AppWin::hWnd );

    ShowWindow( AppWin::hWnd, SW_HIDE );
//    ShowWindow( hWnd, nCmdShow );
//    UpdateWindow( hWnd );

    MSG msg;
```

Page 7 of 8

```
    while( GetMessage( &msg, 0, 0, 0 ) )
    {
        TranslateMessage( &msg );
        DispatchMessage( &msg );
    }

    return msg.wParam;
}


void AppWin::onCommand( HWND hWnd, UINT cmd, HWND, UINT )
{
    switch ( cmd )
    {
     case WM_INIT_DIALOG:
        {
            // create dialog box proc instances for use later
            static FARPROC lpfn = MakeProcInstance( (FARPROC )DlgWin::wndProc, h
  => Instance );

            assert( lpfn );

            DialogBox( hInstance, "TestDialog", hWnd, (DLGPROC )lpfn );

            PostMessage( hWnd, WM_CLOSE, 0, 0 );
        }
        break ;


     default:
        messageHandled = FALSE;
    }
}

bool AppWin::onCreate( HWND hwnd, CREATESTRUCT FAR* )
{
    hTask = GetCurrentTask();

    hWnd = hwnd;

    PostMessage( hwnd, WM_COMMAND, UM_INIT_DIALOG, 0 );

    return TRUE;
}

void AppWin::onDestroy( HWND )
{
    PostQuitMessage( 0 );
}

void AppWin::onClose( HWND hWnd )
{
    DestroyWindow( hWnd );
}


long FAR PASCAL _export AppWin::wndProc( HWND hWnd, UINT message, WPARAM wParam,
  =>  LPARAM lParam )
{
```

```
    messageHandled = TRUE;

    long returnValue = 0;

    switch( message )
    {
        handleMessage( WM_CREATE, hWnd, wParam, lParam, onCreate );
        handleMessage( WM_DESTROY, hWnd, wParam, lParam, onDestroy );
        handleMessage( WM_CLOSE, hWnd, wParam, lParam, onClose );
        handleMessage( WM_COMMAND, hWnd, wParam, lParam, onCommand );

     default:
        messageHandled = FALSE;

        break;
    }

    return( messageHandled
            ? returnValue
            : DefWindowProc( hWnd, message, wParam, lParam ) );
}
```

WTRAIN.CPP 7-22-95 11:35a

Page 8 of 8

```
/*****************************************************************************
        Description

        ordcoll.cpp
        base classes to manage ordered Collection. Like Array collection
        but the items in the arrays follow an ordering principle set up by the
        programmer.

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995

*****************************************************************************/

#include "ordcoll.h"
#include "defs.h"
#include <mem.h>


#if DEBUG || UNIT_1


OCBase::OCBase( int initialsize, int incrementsize )
{
    arraySize = ( initialsize <= 0 ) ? 6 : initialsize + 1;

    incrementSize = ( incrementsize <= 0 ) ? arraySize : incrementsize;

    array = new void*[ arraySize + 1 ] + 1;

    arrayUsed = 0;

    array[ 0 ] = 0;

    *(array - 1) = 0;
}

#endif
#if DEBUG || UNIT_2


void OCBase::add( void* v )
{
    if( arrayUsed == arraySize-1 )
        add( v, arrayUsed );

    else
    {
        array[ arrayUsed +1 ] = array[ arrayUsed ];
        array[ arrayUsed++ ] = v;
    }
}


#endif
#if DEBUG || UNIT_3


void OCBase::add( void* e, int index )
{
    assert( arrayUsed <= arraySize );

    assert( index <= arrayUsed ); //  if( index > arrayUsed ) index=arrayUsed;
```

ORDCOLL.CPP 7-22-95 11:29a

```
    if( arrayUsed == arraySize-1 ) {

        arraySize += incrementSize;

        incrementSize += (incrementSize>>2) + 48/(1 + (47/incrementSize) );

        assert( arraySize+1 < 0x4000 );

        void **newArray = new void*[ arraySize +1 ];

        assert( newArray );

        memcpy( newArray, array-1, (index +1) * sizeof(void *) );

        memcpy( newArray +index +2,
                array +index,
                (arrayUsed -index +1)*sizeof( void* ) );

        delete [] (array -1);

        array = newArray +1;
    }
    else {
        memmove( array +index +1, array +index,
                (arrayUsed -index +1)*sizeof( void* ) );
    }

    *(array +index) = e;

    ++arrayUsed;
}


#endif
#if DEBUG || UNIT_4


void* OCBase::remove( void* e )
{
    int i = find( e );

    return ( i != -1 ) ? removeEntry( i ) : 0;
}


#endif
#if DEBUG || UNIT_5


void* OCBase::removeEntry( int index )
{
    assert( index >=0 );

    if( index < arrayUsed )
    {
        void* rtn = array[ index ];

        memmove( array + index, array + index + 1,
                (arrayUsed - index) * sizeof( void* ) );

        arrayUsed -= 1;

        return rtn;
    }
    else
        return 0;
```

Page 1 of 2

```
}

#endif
#if DEBUG || UNIT_6

int OCBase::find( const void* e ) const
{
    for( int i = 0; i < arrayUsed; ++i )
    {
        if( array[i] == e ) return i;
    }

    return -1;
}

#endif
#if DEBUG || UNIT_7

int OCBase::find( const void* v, CompareOCEntries cmp,
                                int sortedList ) const
{
    assert( cmp );

    int i;

    if( sortedList )
        return find( v, cmp, &i );
    else
    {
        for( i = 0; i < arrayUsed; ++i )
        {
            if( !cmp( v, array[i] ) ) return i;
        }
    }

    return -1;
}

#endif
#if DEBUG || UNIT_8

int OCBase::find( const void* e, CompareOCEntries compare,
                                int* position ) const
{
    // returns -1 if it cannot find the entry
    // if the compare function is given, it does a log2 search
    // if position is given and the entry is not found, position will
    // be the proper position this item should be added at;

    assert( compare );

    int testPosition=0, lowIndex=0, highIndex=count() -1, cmp=-1;

    static int lastFound = -1;

    if( lastFound > highIndex || lastFound < 0 )
        lastFound = ( lowIndex + highIndex + 1 ) / 2;

    while( lowIndex <= highIndex )
    {
```

ORDCOLL.CPP 7-22-95 11:29a

```
        testPosition = lastFound;

        cmp = compare( e, (*this)[ testPosition ] );

        if( cmp == 0 ) break;

        else if( cmp > 0 )
            lowIndex = testPosition + 1;

        else    // cmp < 0
            highIndex = testPosition - 1;

        lastFound = ( lowIndex + highIndex ) / 2;
    }

    if( position ) *position = testPosition + (cmp > 0);

    return (cmp == 0) ? testPosition : -1 ;
}

#endif

#if 0

void OCBase::append( OCBase *given )
{
    assert( given );

    unsigned givenUsed = given->count();
    unsigned newSize = (arrayUsed + givenUsed +5) & 0xfffc;

    assert( newSize < 0x4000 );

    if( arraySize < newSize )
    {
        void **newArray = new void*[ newSize + 1 ];

        memcpy( newArray, array-1, (arrayUsed +1) * sizeof(void *) );

        delete [] (array -1);

        array = newArray +1;

        arraySize = newSize;
    }

    memcpy( array +arrayUsed, given->array, (givenUsed +1) * sizeof(void *) );

    arrayUsed += givenUsed;
}

#endif
```

Page 2 of 2

```
/*****************************************************************************
        Description

        DEQ.cpp
        base classes to manage doubly linked queue.

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995

*****************************************************************************/


#include "deq.h"
#include <stddef.h>
#include <assert.h>


void DLinkBase::addNext( DLinkBase* newItem ) {
    // this is in the list, newItem is inserted at this's next
    assert( newItem != 0 );

    if( (newItem->nextLink=nextLink) != 0 )
        nextLink->prevLink=newItem;

    newItem->prevLink = this;
    nextLink = newItem;
}

void DLinkBase::addPrev( DLinkBase* newItem ) {
    // this is in the list, newItem is inserted at this's prev
    assert( newItem != 0 );

    if( (newItem->prevLink=prevLink) != 0 )
        prevLink->nextLink=newItem;

    newItem->nextLink = this;
    prevLink = newItem;
}

void DLinkBase::remove() {
    //  removes this from linked list
    if( nextLink )
        nextLink->prevLink = prevLink;
    if( prevLink )
        prevLink->nextLink=nextLink;

    nextLink = prevLink = 0;
}


void DeqBase::addFirst( DLinkBase* list ) {
    //  inserts this BELOW the newItem;
    list->setPrev( (DLinkBase*) 0);
    list->setNext(f);
    if( f )
        f->setPrev(list);
    else
        l=list;
    f=list;
}

void DeqBase::addLast( DLinkBase* list ) {
    //  inserts this BELOW the last()
    list->setNext( (DLinkBase*) 0);
                    DEQ.CPP  7-22-95  11:35a
```

```
    list->setPrev(l);
    if( l )
        l->setNext(list);
    else
        f=list;
    l=list;
}

void DeqBase::addNext( DLinkBase* list, DLinkBase* newItem ) {
    if( list==l )
        addLast( newItem );
    else
        list->addNext( newItem );
}

void DeqBase::addPrev( DLinkBase* list, DLinkBase* newItem ) {
    if( list==f )
        addFirst( newItem );
    else
        list->addPrev( newItem );
}

void DeqBase::remove( DLinkBase* list ) {
    //  removes list from DeqBase updating first and last;
    if( list->prev() )
        list->prevLink->nextLink = list->next();
    else
        f=list->next();

    if( list->next() )
        list->nextLink->prevLink = list->prev();
    else
        l = list->prev();

    list->prevLink = list->nextLink = 0;
}

DeqBase *DeqBase::bisect( DLinkBase* cutoff ) {
// this->last() != cutoff, the returned Deq's first()=cutoff

#if DEBUG
    assert( cutoff != 0 );

    for( DLinkBase* p = f; p != 0; p = p->next() )
        if( p == cutoff ) break;

    assert( p == cutoff );
#endif

    DeqBase* newDeq = new DeqBase;

    newDeq->f = cutoff;
    newDeq->l = l;

    l = cutoff->prev();
    if( l )
        l->setNext( 0 );
    else
        f = 0;

    if( cutoff )
        cutoff->setPrev( 0 );

    return newDeq;
}
                    Page 1 of 2
```

```
void DeqBase::merge( DeqBase* endDeq ) {
    // connects the end Deq to the bottom of this
    assert(endDeq->f != 0);
    assert(l != 0);
    endDeq->f->prevLink = l;
    l->nextLink = endDeq->f;
    l = endDeq->l;
    endDeq->removeAll();
}

void DeqBase::sort( DLinkCompare compare ) {
    // a straight insertion sort
    // assume the list are aproximately ordered...
    // then insertion sort should not be that bad
    DLinkBase *ins, *nxt, *list=f;
    if (list==0) return;
        #if DEBUG
    long j;
    assert((j=count()) != 0);
        #endif

    ins=list->next();
    remove(list);

    while (ins != 0) {
        nxt=ins->next();
        remove( ins );
        while( compare( ins, list ) > 0) {
            if( list->next() == 0 ) break;
            else list=list->next();
        }
        while( compare( ins, list ) < 0) {
            if( list->prev() == 0 ) {
                ins->nextLink = list;                    list->prevLink = ins;
                goto insertionDone;
            }
            else list=list->prev();
        }
        list->addNext( ins );
insertionDone:
        ins=nxt;
    }
    nxt=list;

    while (list->prev() != 0)   list=list->prev();
    f=list;

    while (nxt->next() != 0)            nxt=nxt->next();
    l=nxt;

    assert(j==count());
    assert(sorted(compare));
}


#if DEBUG

long DeqBase::count() const {
    long i=0;
    for (DLinkBase *list=f; list; list=list->next()) ++i;
    return i;
}

bool DeqBase::sorted( DLinkCompare compare ) {
    DLinkBase *list=f;
    assert(list != 0);
```

DEQ.CPP 7-22-95 11:35a

```
    int i;
    while (list->next() != 0) {
        i=compare(list,list->next());
        if (i > 0)      return FALSE;
        list=list->next();
    }
    return TRUE;
}

void DeqBase::removeAll()
{
    DLinkBase *d, *nextLink;

    for( d = f; d != 0; d = nextLink )
    {
        nextLink = d->next();

        d->setNext( 0 );
        d->setPrev( 0 );
    }

    f = 0;
    l = 0;
}

bool DeqBase::validateDeq()
{
    if( f == 0 ) return( l == 0 );

    for( DLinkBase* d = f; d != 0; d = d->nextLink )
    {
        if( d == d->nextLink ) return FALSE;

        for( DLinkBase *p = d->prevLink; p != 0; p = p->prevLink )
            if( d == p ) return FALSE;
    }

    for( d = l; d != 0; d = d->prevLink )
    {
        if( d->prevLink == 0 && f != d ) return FALSE;

        if( d == d->prevLink ) return FALSE;
    }

    return TRUE;
}

#endif
```

Page 2 of 2

```
/****************************************************************************
        Description

        hashcsz.cpp
        Hash module

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995
*****************************************************************************/

#include "std.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>

#include "hashcsz.h"


#define errorExit( s )  { printf( s ); exit( 1 ); }
#define LINE_SIZE       256


OneShotHeap StateZip::osHeap( 30 * 1024 );
StateToStateZipHS StateZip::stateToStateZipHS;

// Compare city names and return difference between zips
int CityToZipHS::compare( const SLinkBase* slb1, const SLinkBase* slb2 ) const
{
    assert( slb1 );
    assert( slb2 );

        // Compare city names
    int rtn = strcmp( ((CitySL* )slb1)->name, ((CitySL* )slb2)->name );

    if( rtn == 0 )
    {
        if( ( (CitySL* )slb1 )->zip == WILDZIP ||
            ( (CitySL* )slb2 )->zip == WILDZIP )
            return 0;

        if( ( (CitySL* )slb1 )->zip > ( (CitySL* )slb2 )->zip )
            return 1;

        else if( ( (CitySL* )slb1 )->zip < ( (CitySL* )slb2 )->zip )
            return -1;

        else
            return 0;
    }

    return rtn;
}

// hash the name
unsigned CityToZipHS::keyHash( const SLinkBase* slb ) const
{
    char *s = ( (CitySL* )slb )->name;

    return stringHash( s );
}
```

```
// hash the zip
unsigned ZipToCityHS::keyHash( const SLinkBase* slb ) const
{
    return ((CitySL* )slb)->zip;
}

// compare zips and return difference between city names
int ZipToCityHS::compare( const SLinkBase* slb1, const SLinkBase* slb2 ) const
{
    long rtn = ( (CitySL* )slb1 )->zip - ( (CitySL* )slb2 )->zip;

    if( rtn == 0 )
    {
        if( ((CitySL* )slb1)->name == 0 || ((CitySL* )slb2)->name == 0 )
            return 0;

        return strcmp( ((CitySL* )slb1)->name, ((CitySL* )slb2)->name );
    }
    else if( rtn > 0 )
        return 1;

    else
        return -1;
}

// Hash name
unsigned StateToStateZipHS::keyHash( const SLinkBase* slb ) const
{
    const char *s = ((StateToStateZipSL* )slb)->name;

    return stringHash( s );
}


int StateToStateZipHS::compare( const SLinkBase* slb1, const SLinkBase* slb2 ) c
  => onst
{
   return strcmp( ((StateToStateZipSL* )slb1)->name, ((StateToStateZipSL* )slb2
  => )->name );
}



//////////////////////////////////////////////////////////////////////////
// Class that knows about the zips for each state
//////////////////////////////////////////////////////////////////////////


StateZip::StateZip( const char* nm, const char* l2,
                    const char* sZip, const char* eZip,
                    const char* id )
{
    name = nm;
    twoLetterCode = l2;
    startZip = atol( sZip );
    endZip = atol( eZip );
    upsID = atol( id );
    pure( 0 );
}

// Array that defines the range of zips legal for each state
StateZip usaStates[] =
{
    StateZip( "airport", "AS", "00100", "00299", "00" ),// UPS airport pseudo
    StateZip( "canada", "CN", "00300", "00599", "00" ), // UPS Canada pseudo
    StateZip( "puertorico", "PR", "00600", "00999", "00" ),
StateZip( "virginislands", "VI", "00800", "00899", "00" ),
StateZip( "alabama", "AL", "35000", "36999", "01" ),
StateZip( "alaska", "AK", "99500", "99999", "02" ),
StateZip( "arizona", "AZ", "85000", "86599", "03" ),
StateZip( "arkansas", "AR", "71600", "72999", "04" ),
StateZip( "california", "CA", "90000", "96199", "05" ),
StateZip( "colorado", "CO", "80000", "81699", "06" ),
StateZip( "connecticut", "CT", "06000", "06999", "07" ),
StateZip( "delaware", "DE", "19700", "19999", "08" ),
StateZip( "districtofcolumbia", "DC", "20000", "20599", "09" ),
StateZip( "florida", "FL", "32000", "34999", "10" ),
StateZip( "georgia", "GA", "30000", "31999", "11" ),
StateZip( "hawaii", "HI", "96700", "96899", "12" ),
StateZip( "idaho", "ID", "83200", "83899", "13" ),
StateZip( "illinois", "IL", "60000", "62999", "14" ),
StateZip( "indiana", "IN", "46000", "47999", "15" ),
StateZip( "iowa", "IA", "50000", "52899", "16" ),
StateZip( "kansas", "KS", "66000", "67999", "17" ),
StateZip( "kentucky", "KY", "40000", "42799", "18" ),
StateZip( "louisiana", "LA", "70000", "71499", "19" ),
StateZip( "maine", "ME", "03900", "04999", "20" ),
StateZip( "maryland", "MD", "20600", "21999", "21" ),
StateZip( "massachusetts", "MA", "01000", "02799", "22" ),
StateZip( "michigan", "MI", "48000", "49999", "23" ),
StateZip( "minnesota", "MN", "55000", "56799", "24" ),
StateZip( "mississippi", "MS", "38600", "39799", "25" ),
StateZip( "missouri", "MO", "63000", "65899", "26" ),
StateZip( "montana", "MT", "59000", "59999", "27" ),
StateZip( "nebraska", "NE", "68000", "69399", "28" ),
StateZip( "nevada", "NV", "89000", "89899", "29" ),
StateZip( "newhampshire", "NH", "03000", "03899", "30" ),
StateZip( "newjersey", "NJ", "07000", "08999", "31" ),
StateZip( "newmexico", "NM", "87000", "88499", "32" ),
StateZip( "newyork", "NY", "10000", "14999", "33" ),
StateZip( "northcarolina", "NC", "27000", "28999", "34" ),
StateZip( "northdakota", "ND", "58000", "58899", "35" ),
StateZip( "ohio", "OH", "43000", "45899", "36" ),
StateZip( "oklahoma", "OK", "73000", "74999", "37" ),
StateZip( "oregon", "OR", "97000", "97999", "38" ),
StateZip( "pennsylvania", "PA", "15000", "19699", "39" ),
StateZip( "rhodeisland", "RI", "02800", "02999", "40" ),
StateZip( "southcarolina", "SC", "29000", "29999", "41" ),
StateZip( "southdakota", "SD", "57000", "57999", "42" ),
StateZip( "tennessee", "TN", "37000", "38599", "43" ),
StateZip( "texas", "TX", "75000", "79999", "44" ),
StateZip( "utah", "UT", "84000", "84799", "45" ),
StateZip( "vermont", "VT", "05000", "05999", "46" ),
StateZip( "virginia", "VA", "22000", "24699", "47" ),
StateZip( "washington", "WA", "98000", "99499", "48" ),
StateZip( "westvirginia", "WV", "24700", "26899", "49" ),
StateZip( "wisconsin", "WI", "53000", "54999", "50" ),
StateZip( "wyoming", "WY", "82000", "83199", "51" ),

// holes: illegal zips
StateZip( "hole1", "Q1", "96900", "96999", "52" ),
StateZip( "hole2", "Q2", "96200", "96699", "53" ),
StateZip( "hole3", "Q3", "89900", "89999", "54" ),
StateZip( "hole4", "Q4", "88500", "88999", "55" ),
StateZip( "hole5", "Q5", "86600", "86999", "56" ),
StateZip( "hole6", "Q6", "84800", "84999", "57" ),
StateZip( "hole7", "Q7", "83900", "83999", "58" ),
StateZip( "hole8", "Q8", "81700", "81999", "59" ),
StateZip( "hole9", "Q9", "71500", "71599", "60" ),
StateZip( "holea", "QA", "69400", "69999", "61" ),
StateZip( "holeb", "QB", "65900", "65999", "62" ),
StateZip( "holec", "QC", "58900", "58999", "63" ),
```

NASHC82.CPP 7-22-95 10:23a

```
    StateZip( "holed", "QD", "56800", "56999", "64" ),
    StateZip( "holee", "QE", "52900", "52999", "65" ),
    StateZip( "holef", "QF", "45900", "45999", "66" ),
    StateZip( "holeg", "QG", "42800", "42999", "67" ),
    StateZip( "holeh", "QH", "39800", "39999", "68" ),
    StateZip( "holei", "QI", "26900", "26999", "69" ),
    StateZip( "holej", "QJ", "09000", "09999", "70" ),
    StateZip( "holel", "QL", "00000", "00099", "72" ),
);

#define NUMSTATEZIPS ( sizeof( usaStates )/sizeof( StateZip ) )

unsigned char StateZip::zipToUsaStateZip[ 1000 ];
unsigned short* StateZip::firstDigitBitField;
unsigned short* StateZip::firstDigitPairBitField;


// initialize StateZip
void StateZip::initZipToUsaStateZip()
{
    if( zipToUsaStateZip[ 0 ] != 0 )
        return;

    firstDigitBitField = new unsigned short[ NUMSTATEZIPS ];
    memset( firstDigitBitField, 0, NUMSTATEZIPS * sizeof( short ) );

    firstDigitPairBitField = new unsigned short[ NUMSTATEZIPS * 7 ];
    memset( firstDigitPairBitField, 0, NUMSTATEZIPS * 7 * sizeof( short ) );

        // NUMSTATEZIPS number of states in usaState array
    for( int i = 0; i < NUMSTATEZIPS; ++i )
    {
        StateZip& sz = usaStates[ i ];

        long zip = sz.startZip;
        // for each state define a bit code that defines whether the first or se
=> con
        // digit of a zip is legal for that state
        for( zip -= zip%100; zip <= sz.endZip; zip += 100 )
        {
            if( zip == 800 && *(sz.stateName()) == 'p' )        // puertorico
                continue;

            zipToUsaStateZip[ zip/100 ] = i;

            firstDigitBitField[ i ] |=  0x0001 << (zip/10000);

            firstDigitPairBitField[ i*7 + ((zip/1000)>>4) ] |= 0x0001 << ((zip/1
=> 000) & 0x0f);
        }
    }

        // for each state there is a hashed array on the state name of with
        // linked zips
    for( i=0; i<NUMSTATEZIPS; ++i )
    {
        stateToStateZipHS.add( new( &osHeap ) StateToStateZipSL( i, usaStates[ i
=>  ].name ) );
    }
}

// find state for a certain zip
StateZip* StateZip::findStateZip( long zip )
{
    return &usaStates[ zipToUsaStateZip[ zip/100 ] ];
}
```

Page 2 of 5

```
// find zips for a certain state name
StateZip* StateZip::findStateZip( char* stateName )
{
    StateToStateZipSL tmp( 0, stateName );

    StateToStateZipSL *fnd = stateToStateZipHS.find( &tmp );

    if( fnd )
        return &usaStates[ fnd->usaIndex ];

    else
        return 0;
}

StateToStateZipSL* StateZip::findStateToStateZipSL( char* stateName )
{
    StateToStateZipSL tmp( 0, stateName );

    return stateToStateZipHS.find( &tmp );
}

bool StateZip::isFirstDigitLegal( int dgt1, int stateId )
{
    return 0 != (firstDigitBitField[ stateId ] & (0x0001 << dgt1));
}

bool StateZip::areFirstTwoDigitsLegal( int dgt1, int dgt2, int stateId )
{
    int index = dgt1 * 10 + dgt2;

    return 0 != ( firstDigitPairBitField[ stateId*7 + (index>>4) ]
                & (0x0001 << (index & 0x0f)) );
}

bool StateZip::areFirstThreeDigitsLegal( int dgt1, int dgt2, int dgt3, int state
  » Id )
{
    return stateId == zipToUsaStateZip( dgt1 * 100 + dgt2 * 10 + dgt3 );
}

// find matching zips, state and cities, given a zip choice list and city
// choice list
void StateZip::zipToStateCities( ZipHypothesis* zHypo, AC< CityHypothesis > *cit
  » yHypo )
{
    long zip = zHypo->value();

    StateZip* sz = StateZip::findStateZip( zip );

    bool zipAgrees = !strcmp( sz->name, zHypo->stateName() );

    CitySL test( zip, 0 );

    for( CitySL* csl = sz->z2c.find( &test ); csl; csl = csl->next() )
    {
        if( zip == csl->zip )
        {
            for( int i=cityHypo->count(); i--; )
            {
                if( !strcmp( csl->name, (*cityHypo)[ i ].cityName ) )
                    goto endOfFind;
            }

            CityHypothesis* ch = &( (*cityHypo)[ cityHypo->makeRoom( 1 ) ] );
            ch->zh = zHypo;
            ch->cityName = csl->name;
            ch->sz = sz;
            ch->zipAgreesWithState = zipAgrees;
        }
endOfFind:

    }
}
// Get zips for cities
char* StateZip::cityToZips( char* city, AC< long > *zips )
{
    CitySL test( WILDZIP, city );

    char* rtn=0;

    for( CitySL* csl = c2z.find( &test ); csl; csl = csl->next() )
    {
        if( !strcmp( city, csl->name ) )
        {
            zips->add( csl->zip );
            rtn = csl->name;
        }
    }
    return rtn;
}

// parse lines of address list file
static bool parseLine( char* line, long* z, char** stateName, char** cityName )
{
    char *s = line;

    while( *s && isdigit( *s ) ) ++s;

    if( *s == '\0' || s == line )
        return FALSE;

    *s++ = '\0';

    *z = atol( line );

    *stateName = s;

    while( *s && isspace( *s ) ) ++s;

    if( *s == '\0' )
        return FALSE;

    *s++ = '\0';

    for( *cityName = s; *s; ++s )
    {
        if( *s == '\r' || *s=='\n' )
        {
            *s = '\0';
            break;
        }
    }

    return TRUE;
}

// Initialize hash tables
int StateZip::initHSs( char* filename )
{
    assert( filename );
```

HASHCSZ.CPP 7-22-95 10:23a

Page 3 of 5

```
    FILE *dataFile = fopen( filename, "r" );

    if( dataFile == 0 )
        return 0;

    char line[ LINE_SIZE ];

    StateZip *sz = 0;

    while( fgets( line, sizeof( line ), dataFile ) != 0 )
    {
        int readSize = strlen( line );

        if( readSize < 5 ) continue;

        assert( readSize < sizeof( line ) );

        long zip;

        char *cityName, *stateName;

        if( parseLine( line, &zip, &stateName, &cityName ) )
        {
            if( sz == 0 )
            {
                sz = findStateZip( zip );
            }

            CitySL test( zip, cityName );

            char* cty = 0;

            for( CitySL *csl=sz->c2z.firstCollision( &test ); csl; csl=csl->next
=> () )
            {
                if( !strcmp( csl->name, cityName ) )
                {
                    cty = csl->name;

                    if( zip == csl->zip )
                        break;
                }
            }

            if( csl == 0 )
            {
                if( cty == 0 )
                {
                    cty = new( &osHeap ) char[ strlen( cityName ) +1 ];

                    strcpy( cty, cityName );
                }

                sz->c2z.add( new( &osHeap )  CitySL( zip, cty ) );
                sz->z2c.add( new( &osHeap )  CitySL( zip, cty ) );
            }
        }
        else
            break;
    }

    fclose( dataFile );

    return sz ? sz->c2z.count() : 0;
}
```

MASHCSZ.CPP 7-22-95 10:23a

```
#if 0
// Test main for this module
void main( int, char** )
{
    StateZip::initZipToUsaStateZip();

    usaStates[ 6 ].initHSs( "../wcpp/zip/zip.05" );

    usaStates[ 7 ].initHSs( "../wcpp/zip/zip.06" );

    char buf[ LINE_SIZE ];

    OC< char > cityHypo;
    AC< long > zips;

    for(;;)
    {
        fgets( buf, LINE_SIZE, stdin );

        char *s = buf;

        int i = strlen( buf );

        if( i )
            buf[ i-1 ] = '\0';

        if( *s == 'q' )
            return;

        if( isdigit( *s ) )
        {
            long zip = atol( s );

            StateZip* sz = StateZip::findStateZip( zip );

            cityHypo.removeAll();

            char *state;

            sz->zipToStateCities( zip, &state, &cityHypo );

            for( int i=cityHypo.count(); i; )
                printf( "city is %s\n", cityHypo[ --i ] );

            if( cityHypo.count() == 0 )
                printf( "none found\n" );
        }
        else
        {
/*          while( *s != ' ' )
            {
                if( *s == '\0' )
                    return;
            }
            while( *s == ' ' )
                *s++ = '\0';

            zips.removeAll();

            StateZip* sz = csl->cityToZips( s, &zips );

*/          for( int i=zips.count(); i; )
                printf( "zip is %ld\n", zips[ --i ] );
```

Page 4 of 5

```
            if( zips.count() == 0 )
*/
                printf( "none found\n" );
        }
    }
}
#endif

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
```

```
/***************************************************************************
        Description

        memuse.cpp
        Module taht computes memory usage during the operation of the applicatio
  => n.

    Copyright (c) 1991-1995 by Dragon Systems, Inc.

    Author: Greg Gadbois
    Created: 1991 -1995
***************************************************************************/

#include <toolhelp.h>
#include <string.h>

const char *moduleList[] =
{
    "WAPP",
    "DRAGDEV",
    "DRAGACPA",
    "DRAGMEDI",
    "DRAGON",
    0
};

// Compute the memory consumption by walking the global heap.
// Count every non-discardable block which belongs to our EXE or DLLs.

long computeMemoryUse()
{
    long totalMemory = 0;

    // load global information
    GLOBALINFO  gInfo;
    gInfo.dwSize = sizeof( GLOBALINFO );

    GlobalInfo( &gInfo );

    // loop through the global heap
    GLOBALENTRY gEntry;
    gEntry.dwSize = sizeof( GLOBALENTRY );

    MODULEENTRY mEntry;
    mEntry.dwSize = sizeof( MODULEENTRY );

    TASKENTRY taskEntry;
    taskEntry.dwSize = sizeof( TASKENTRY );

    for( BOOL bHasEntry = GlobalFirst( &gEntry, GLOBAL_ALL );
         bHasEntry;
         bHasEntry = GlobalNext( &gEntry, GLOBAL_ALL ) )
    {
        // ignore free blocks
        if( !gEntry.hOwner )    continue;

        // ignore the following types of data blocks
        switch( gEntry.wType )
        {
         case GT_CODE:
         case GT_FREE:
         case GT_INTERNAL:
         case GT_SENTINEL:
         case GT_BURGERMASTER:
            continue;
```

MEMUSE.CPP 7-22-95 10:26a

```
         default:
            break;
        }

        if( 0 == ModuleFindHandle( &mEntry, (HMODULE )gEntry.hOwner ) )
        {
            if( TaskFindHandle( &taskEntry, (HTASK )gEntry.hOwner ) )
                strcpy( mEntry.szModule, taskEntry.szModule );

            else
                continue;
        }

        for( const char** module = moduleList; *module; ++module )
        {
            if( 0 == stricmp( *module, mEntry.szModule ) )
            {
                totalMemory += gEntry.dwBlockSize;

                if( *module == moduleList[ 0 ] )
                    MessageBox( 0, "in Memory", "found WAPP", MB_OK | MB_ICONEXC
=> LAMATION );

                break;
            }
        }
    }

    return totalMemory;
}
```

Page 1 of 1

```
/* ARRCOLL.H */

/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef arrcoll_h
#define arrcoll_h

#include <assert.h>
#include <stdlib.h>
#include <string.h>
#include "defs.h"


typedef int (*CompareACEntries)( const void* given, const void* test);

class ACBase
{
 public:
    // constructors
    ACBase( size_t entrysize, unsigned initialsize = 5, unsigned incrementsize =
  => 5 );

    // destructor
    ~ACBase();          // does not call element destructors

    int count() const;
    bool isEmpty() const;

    int increment() const;
    int increment( int i );

    void* first();
    void* last();
    void* next( char* p );
    void* prev( char* p );


    char& operator[]( const int i ) const;

    int add( const char& item, int index=-1 );
        // index == -1 is a wildcard for end of file
        // returns a proper index or -1 on failure

    int makeRoom( int numToAdd=1, int index=-1 );
        // allocates a block of elements starting at index
        // (-1 is a wildcard meaning at the end the array)
        // returns the index of the first item or -1 on failure

    void remove( char& v );
    void removeEntry( unsigned i );
    void removeAll();

    int find( const char& v ) const;
    int find( const char& v, CompareACEntries cmp, int sortedList=0 ) const;
    int find( const char& v, CompareACEntries cmp, int* position ) const;
        // returns -1 if it cannot find the entry
        // if only a cmp function is given it is assumed it is an unsorted
        // list... we do a stupid exhastive search
        // if the compare function is given, and sortedList is TRUE we do
        // a log2 search
        // if a position is given we do a log2 search and
        // if position is given and the entry is not found, position will
        // be the proper position this item should be added at;

    int indexOf( const char& v ) const;
```

ARRCOLL.H 7-22-95 2:12p

```
    void sort( CompareACEntries cmp );
    void merge( ACBase* given );

 protected:
    unsigned arraySize;
    unsigned arrayUsed;
    unsigned entrySize;
    unsigned incrementSize;

    char* array;
    void* lastEntry() const;
};


inline ACBase::~ACBase()
{
    delete [] array;
}


inline void* ACBase::lastEntry() const
{
    return array + (arrayUsed -1) * entrySize;
}


inline int ACBase::count() const
{
    return arrayUsed;
}

inline bool ACBase::isEmpty() const
{
    return( count() == 0 );
}

inline int ACBase::increment() const
{
    return incrementSize;
}

inline int ACBase::increment( int i )
{
    int rtn = incrementSize;
    incrementSize = i;
    return rtn;
}

inline void* ACBase::first()
{
    return( arrayUsed > 0 ? array : 0 );
}

inline void* ACBase::last()
{
    return( arrayUsed > 0 ? lastEntry() : 0 );
}

inline void* ACBase::next( char* p )
{
    assert( p>=array );
    return( p < lastEntry() ? p+entrySize : 0 );
}

inline void* ACBase::prev( char* p )
{
```

Page 1 of 2

```
    assert( p<=lastEntry() );
    return( p > array ? p-entrySize : 0 );
}

inline char& ACBase::operator[]( const int i ) const
{
    assert( (unsigned )i < (unsigned )count());
    return* ( array + i*entrySize );
}

inline void ACBase::remove( char& e )
{
    removeEntry( find( e ) );
}

inline void ACBase::removeAll()
{
    arrayUsed = 0;
}

inline void ACBase::sort( CompareACEntries compare )
{
    qsort( (void*) array, arrayUsed, entrySize, compare );
}


template <class T>
class AC : public ACBase
{
public:
    AC( unsigned initialsize = 5, unsigned incrementsize = 5 ) : ACBase( sizeof(
  => T ), initialsize, incrementsize ) { }

    T* first() { return (T* ) ACBase::first(); }
    T* last() { return (T* ) ACBase::last(); }

    T* next( T* e )
    {
        assert( e >= (T* )array );
        return( e+1 < ((T* )array) +arrayUsed ? e+1 : 0 );
    }
    T* prev( T* e )
    {
        assert( e < ((T* )array) + arrayUsed );
        return( e > (T* )array ? e-1 : 0 );
    }

    T& operator[]( int i ) const { return (T& ) ((T*)array)[ i ]; }
//    const T* operator[]( int i ) const { return (const T* ) ACBase::operator[]
  => (i); }

    void add( const T& e, int index=-1 ) { ACBase::add( (const char& )e, index )
  => ; }

    void remove( const T& e ) { ACBase::remove( (char& )e ); }

    int find( const T& e ) const { return ACBase::find( (const char& ) e); }
    int find( const T& e, CompareACEntries cmp, int sortedList=0 ) const {
        return ACBase::find( (const char& ) e, cmp, sortedList );
    }
    int find( const T& e, CompareACEntries cmp, int* position ) const {
        return ACBase::find( (const char& ) e, cmp, position );
    }
    int indexOf( const T& e ) const {
        return ACBase::indexOf( (const char& ) e );
    }
```

ARRCOLL.H 7-22-95 2:12p

```
    void merge( AC<T>* given );
};

#endif // arrcoll_h
```

Page 2 of 2

```
/* DEQ.H
 => */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef deq_h
#define deq_h

typedef int bool;

#define TRUE        1
#define FALSE       0


////////////////////////////////////////////////////////////////////
// DLink Template
////////////////////////////////////////////////////////////////////

class DLinkBase
{
private:
    DLinkBase *prevLink, *nextLink;
protected:
    void setPrev( DLinkBase* list ) { prevLink=list; }
    void setNext( DLinkBase* list ) { nextLink=list; }

    void addNext( DLinkBase* );
    void addPrev( DLinkBase* );
    void remove();

public:
    DLinkBase() { prevLink = nextLink = 0; }

    DLinkBase *prev() const { return prevLink; }
    DLinkBase *next() const { return nextLink; }

    friend class DeqBase;
    friend class HDBase;
};


template <class T>
class DLink : public DLinkBase
{
protected:
    void setPrev( T* list ) { DLinkBase::setPrev( (DLinkBase*) list ); }
    void setNext( T* list ) { DLinkBase::setNext( (DLinkBase*) list ); }

    void addNext( T* list ) { DLinkBase::addNext( (DLinkBase*) list ); }
    void addPrev( T* list ) { DLinkBase::addPrev( (DLinkBase*) list ); }
    // void remove() { DLinkBase::remove(); }

public:
    T *prev() const { return (T*) DLinkBase::prev(); }
    T *next() const { return (T*) DLinkBase::next(); }
};


typedef int (*DLinkCompare)( DLinkBase*, DLinkBase* );
//      compare(DLink *a, DLink *b) ==> (a - b)


////////////////////////////////////////////////////////////////////
// Deq Template
////////////////////////////////////////////////////////////////////
```

DEQ.H 7-22-95 2:12p

```
class DeqBase {
protected:
    DLinkBase *f, *l;
public:
    DeqBase() { f=l=0; }

    // DLinkBase manipulation routines
    DLinkBase* first() const { return f; }
    DLinkBase* last() const { return l; }

    void addFirst( DLinkBase* );
    void addLast( DLinkBase* );
    void addNext( DLinkBase* list, DLinkBase* newItem );
    void addPrev( DLinkBase* list, DLinkBase* newItem );

    void remove( DLinkBase* list );

            #if DEBUG
    void removeAll();
            #else
    void removeAll() { f=l=0; }
            #endif

    bool isEmpty() const        { return ( f == 0 ); }

    // Deq manipulation routines
    DeqBase *bisect( DLinkBase* cutoff );
    // this->last() != cutoff, the returned Deq's first()=cutoff
    void merge( DeqBase* nextDeq );
    // connects the given Deq to the end of this
    // nextDeq->removeAll()'d at the end of merge
    void sort( DLinkCompare );

            #if DEBUG
    bool contains( DLinkBase* );
    bool sorted( DLinkCompare );
    long count() const;
    ~DeqBase() { removeAll(); }
    bool validateDeq();
            #endif
};


template <class T>
class Deq : public DeqBase
{
public:
    // DLinkBase manipulation routines
    T* first() const { return (T*) DeqBase::first(); }
    T* last() const { return (T*) DeqBase::last(); }

    void addFirst( T* list ) { DeqBase::addFirst( (DLinkBase*) list ); }
    void addLast( T* list ) { DeqBase::addLast( (DLinkBase*) list ); }
    void addNext( T* list, T* newItem ) { DeqBase::addNext( (DLinkBase*) list, (
  => DLinkBase*) newItem ); }
    void addPrev( T* list, T* newItem ) { DeqBase::addPrev( (DLinkBase*) list, (
  => DLinkBase*) newItem ); }

    void remove( T* list ) { DeqBase::remove( (DLinkBase*) list ); }

    // Deq manipulation routines
    Deq<T>* bisect( T* cutoff ) { return (Deq<T>* )DeqBase::bisect( (DLinkBase*)
  => cutoff ); }
    // this->last() != cutoff, the returned Deq's first()==cutoff
    void merge( Deq<T>* nextDeq ) { DeqBase::merge( (DeqBase*) nextDeq ); }
```

Page 1 of 2

```
    // connects the given Deq to the end of this
    // nextDeq->removeAll()'d at the end of merge
                #if DEBUG
    bool contains( T* list ) ( return DeqBase::contains( (DLinkBase*) list ); )
                #endif
);

#endif /* deq_h */
```

DEQ.H 7-22-95 2:12p

Page 2 of 2

```
/************************************************************************

    MODULE: deamon.h

    PURPOSE: include file for communciations module.

    FUNCTIONS:

    COMMENTS:

Author:   Hal Strausberg     Written:   March 28, 1994

Revisions  (latest first):

Date:          Author:        Change:
-----          -------        -------



  Copyright (C) 1994 Dragon Systems, Inc.

*************************************************************************/
#ifndef _DEAMONH_
#define _DEAMONH_

#include <windowsx.h>
#include <string.h>


// COMM Window Messages
#define UM_COMM_CHECK_MSG        WM_USER+111
#define UM_COMM_SEND_SYNC_MSG    WM_USER+112

//---------------------------------------------------------------
//  comm defines
//---------------------------------------------------------------
#define HCOM HANDLE

/* misc. constants */
#define NFG               (-1)
#define XON               17
#define XOFF              19
#define EVENT_MASK        0x3FF    /* all bits on */
#define INBUFSIZE         2048     /* Windows COM input buffer size */
#define OUTBUFSIZE        2048     /* Windows COM output buffer size */
#define TEMPBUFSIZE       256      /* temporary IO buffer size */
#define NOCHAR_TIMEOUT    200      /* need to send sync message */

// dialogs
#define TIMER_EVENT          666
#define TIMER_INTERVAL       50



// data structures
/* parameter setting struct */
struct CommPars
{
    WORD port;      /* port number (1-n) */
    int baud;
    char parity;
    char databits;
    char stopbits;
```

DEAMON.H 7-22-95 2:08p

```
    int breaklen;
    unsigned        /* flags... */
    inx:1,      /* input xon/xoff */
    outx:1,     /* output xon/xoff */
    cts:1,      /* cts flow ctrl */
    dsr:1,      /* dsr flow ctrl */
    dtr:1,      /* dtr flow ctrl */
    rts:1,      /* rts flow ctrl */
    unused:10;

    CommPars(){ }

    CommPars( int pt, int bd, int pr, int dt, int st )
    {
        port = pt;       baud = bd;      parity = pr;    databits = dt;  stopbits
  => = st;
        breaklen = 300; inx = FALSE;     outx = FALSE;
        cts = FALSE;    dsr = FALSE;     dtr = FALSE;    rts = FALSE;
    }
};


//---------------------------------------------------------------
//                       Classes
//---------------------------------------------------------------

class Comm;

struct CommGlobals
{
};

struct CommPrimitive
{
    int connectId;                         /* ID for an open connection */
    int err;                      /* code from last erroneous operation */
    WORD wTimerID;
    CommPars params;              /* settable parameters */
    CommPars oldparams;           /* saved copy for temporary changes */
    BYTE xrBuffer[INBUFSIZE];     /* received block buffer */
    BYTE xBuffer[OUTBUFSIZE];     /* received block buffer */

    CommPrimitive()
    {
        memset( (LPVOID )this, (BYTE )0, (WORD )sizeof( CommPrimitive ) );

        connectId = NFG;
    }
};

class Comm : public CommPrimitive
{
 protected:
    // globals:
    static Comm* com1;
    static Comm* com2;
    static Comm* com3;
    static Comm* com4;

    // the needed Callbacks...
    static void CALLBACK _export com1TimerProc( HWND hwnd, UINT msg, UINT idTime
  => r, DWORD dwTime );
    static void CALLBACK _export com2TimerProc( HWND hwnd, UINT msg, UINT idTime
  => r, DWORD dwTime );
    static void CALLBACK _export com3TimerProc( HWND hwnd, UINT msg, UINT idTime
  => r, DWORD dwTime );
```

Page 1 of 2

```
   static void CALLBACK _export comTimerProc( HWND hwnd, UINT msg, UINT idTime
»> r, DWORD dwTime );

   static CommPars defaultPars;

   // protected data:
   Comm** global;

   HWND hWnd;           // the hWnd that processes comm events
   BOOL needResync;
   BOOL outGoing;
   int  curNoChar;
   int  index;          // index into the xrBuffer for putting new characters

   // protected service functions:
   void timerProcPrimitive();


 public:

   Comm( HWND, CommPars* =0 );          // hwnd that processes CommMessages
   ~Comm();

   BOOL comConnect();
   BOOL comDisconnect();

   BOOL comReset();

   int comRead();
   int comRead( LPSTR s, int max ) { return ReadComm( connectId, s, max ); }

   int comWrite();
   int comWrite( LPSTR lpStr, int len );

   void comSendBreak( int len );
   BOOL comClearLine( DWORD dwQuiet, DWORD dwMax );

   int comGetParams( CommPars* lpParams );
   BOOL comSetParams( CommPars* lpParams );

   void comGetWindowsParams( CommPars* lpPrms, DCB far *lpDCB );

   void comGetQueueCounts( LPINT lpRxq, LPINT lpTxq );
   void comFlushQueues();

   int comDoOutput( LPSTR lpStr, int len );
   int comCheckOutputStatus();

   int error() { return err; }

   // functions to take over...
   BOOL checkMessage();
   BOOL sendSyncMsg();
};

#endif
```

DEAMON.H 7-22-95 2:08p

Page 2 of 2

```
/* HASHCSZ.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/


#ifndef hashcsz_h
#define hashcsz_h

#include "ordcoll.h"
#include "arrcoll.h"
#include "oneshot.h"
#include "stack.h"
#include "dragcpp.h"


class CityHypothesis;

// public class that contains phrase, spelling, index to the zip
// starting from beginning of phrase, and an index for the spelling

struct ZipResult
{
        // returned by recognizer, example: phrase "blue 0 1 9 1 3 massachusetts
  »> "
    char phrase[ UTT_PROMPT_LENGTH ];
        // same as phrase, but digits have NO spaces
    char spelling[ UTT_PROMPT_LENGTH ];

    short zipStateSpellingIndex;
    short stateSpellingIndex;

        // returns pointer to zip code and following state name
    char* zipStateName() { return spelling + zipStateSpellingIndex; }
    char* stateName()  { return spelling + stateSpellingIndex; }

    ZipResult() {};
    ZipResult( ZipResult& );

    int initFromPhrase();
    int initFromSpelling();
    long value() { return atol( zipStateName() ); }

    bool hasIntegrity() {
        return( (unsigned short) zipStateSpellingIndex < UTT_PROMPT_LENGTH &&
                (unsigned short).stateSpellingIndex < UTT_PROMPT_LENGTH );
    }
};

// zipHypothesis also contains SD_WORD handle for the recognized word
// ziHypothesis inherits from ZipResult
class ZipHypothesis : public ZipResult
{
 public:
    bool isNewWord;
    SD_WORD wordId;

    CityHypothesis* ch;

    ZipHypothesis( ZipHypothesis& );
    ZipHypothesis( Recognizer*, ZipResult& );
    ZipHypothesis( Recognizer*, long, ZipResult& );
    ZipHypothesis() { };
};

// Array of ZipHypotheses
class ZipHypoAC : public AC< ZipHypothesis >
{
 public:
    ZipHypoAC( unsigned initialsize = 5, unsigned incrementsize = 5 )
    : AC< ZipHypothesis >( initialsize, incrementsize ) { }

    void cleanUpNewWords( Recognizer* );
};


/////////////////////////////////////////////////////////////////////////
// This class is used by the stack decoder to generate alternative
// choices to the one zip returned by the continuous recognizer
/////////////////////////////////////////////////////////////////////////


class Hypothesis
{
 protected:
        // Matrix that contains log values for degree of confusablity
        // between each digit pair
    static int confusionScore[ 100 ];

 public:
        // Memory manager
    static OneShotHeap heap;

        // array that keeps track which digit in the zip we are dealing with
    int wordHistoryArray[ 5 ];
    int* data;

        long currentScore;
        // index into wordHistoryArray
    long nextWordIndex;

        // Constructor
    Hypothesis() { currentScore = 1000; nextWordIndex = 0; data = 0; }

    Hypothesis( Hypothesis& h ) {
        currentScore = h.currentScore;
        nextWordIndex = h.nextWordIndex;
        data = h.data;
        memcpy( wordHistoryArray, h.wordHistoryArray, sizeof( wordHistoryArray )
  »  );
    }

        // Generate a new hypothesis by taking the top from the stack
        // and loking at the next digit
    void propagate( PriorityQue<Hypothesis>* hpq, int stateId );

    long value() {
        return( wordHistoryArray[ 0 ] * 10000L + wordHistoryArray[ 1 ] * 1000L
                + wordHistoryArray[ 2 ] * 100L + wordHistoryArray[ 3 ] * 10L
                + wordHistoryArray[ 4 ] );
    }
};

// Heavy duty function that fills up ZipHypoAC with zip alternatives
// Each alternative has a SD_WORD handle
extern int buildHypothesis( Recognizer*, ZipHypoAC*, ZipResult&, int );


class StateZip;

// public class that knows all relevant info about a recognized city name
// analogous to ZipHypothesis
struct CityHypothesis
{
    ZipHypothesis* zh;
    char* cityName;
    StateZip* sz;
    SD_WORD wordId;
    bool zipAgreesWithState;
};

/////////////////////////////////////////////////////////////////////////////
//  Hashed city->zip and zip->city translations
/////////////////////////////////////////////////////////////////////////////


#define WILDZIP         -1L

// Singly linked list of cities
// Each city knows its zip and has a pointer to its name

class CitySL : public SLink< CitySL >
{
 public:
    long zip;
    char *name;

    CitySL( long z, char* n ) {
        zip = z; name = n;
    }

    friend class CityToZipHS;
    friend class ZipToCityHS;
};

// hashed array of lists of cities. The hashing is done on the zip
class CityToZipHS : public HS< CitySL >
{
 protected:

    unsigned keyHash( const SLinkBase* ) const;

 public:
    CityToZipHS( unsigned maxSize = 8192, unsigned initialSize = 2 )
    : HS< CitySL >( maxSize, initialSize ) { }

    int compare( const SLinkBase*, const SLinkBase* ) const;
};

// hashed array of lists of cities. The hashing is done on the city name
class ZipToCityHS : public HS< CitySL >
{
 protected:

    unsigned keyHash( const SLinkBase* ) const;

 public:
    ZipToCityHS( unsigned maxSize = 8192, unsigned initialSize = 2 )
    : HS< CitySL >( maxSize, initialSize ) { }

    int compare( const SLinkBase*, const SLinkBase* ) const;
};

// linked list of State name and their index into the class StateZip
class StateToStateZipSL : public SLink< StateToStateZipSL >
{
 public:
    int usaIndex;
    const char *name;
```

MASHCS2.H 7-22-95 2:12p

```
    StateToStateZipSL( int i, const char* n ) {
        usaIndex = i; name = n;
    }
};


//Hashed array of linked states
class StateToStateZipHS : public HS< StateToStateZipSL >
{
 protected:
    unsigned keyHash( const SLinkBase* ) const;

 public:
    StateToStateZipHS( unsigned maxSize = 128, unsigned initialSize = 128 )
    : HS< StateToStateZipSL >( maxSize, initialSize ) { }

    int compare( const SLinkBase*, const SLinkBase* ) const;
};



class StateZip
{
 protected:
        // memory manager
    static OneShotHeap osHeap;

        // hashed state names
    static StateToStateZipHS stateToStateZipHS;
    static unsigned char zipToUsaStateZip[ 1000 ];
    static unsigned short* firstDigitBitField;
    static unsigned short* firstDigitPairBitField;

    const char* name;
    const char *twoLetterCode;
    long startZip, endZip;
    int upsID;
    bool isPureState;
    CityToZipHS c2z;
    ZipToCityHS z2c;

 public:
    StateZip( const char*, const char*, const char*, const char*, const char* );

    const char* stateName() { return name; }
    char* cityToZips( char* city, AC< long > *zips );
    bool isPure() { return isPureState; }
    void pure( bool trueFalse ) { isPureState = trueFalse; }

    static int initHSs( char* filename );
    static void initZipToUsaStateZip();
        // get state name for zip
    static StateZip* findStateZip( long zip );
        // get zips for state name
    static StateZip* findStateZip( char* stateName );
        //
    static StateToStateZipSL* findStateToStateZipSL( char* stateName );
    static void zipToStateCities( ZipHypothesis* , AC< CityHypothesis >* );

    static const char* findStateName( long zip )
    {
        StateZip* sz = findStateZip( zip );    return( sz ? sz->name : "" );
    }
        // validate first digit as legal for that state
```

Page 2 of 3

```
    static bool isFirstDigitLegal( int dgt1, int stateId );
        // validate first 2 digit as legal for that state
        static bool areFirstTwoDigitsLegal( int dgt1, int dgt2, int stateId );
        // validate first 3 digit as legal for that state
        static bool areFirstThreeDigitsLegal( int dgt1, int dgt2, int dgt3, int
 => stateId );
        // Not used
    static bool areFirstFiveDigitsLegal( int dgt1, int dgt2, int dgt3,
                                         int dgt4, int dgt5, int stateId );
};

#endif
```

```
/* DRAGCPP.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef dragcpp_h
#define dragcpp_h

//#include <std.h>
#include <stdio.h>
#include <assert.h>
#include <windowsx.h>
#include "sdapi.h"
#include "sdapiinh.h"

#include "deq.h"
#include "arrcoll.h"


#define TRUE    1
#define FALSE   0
#define YES     1
#define NO      0

#define MAX_RECOG_CHOICES 10
#define MAX_TEXT_PAR 64
#define UTT_PROMPT_LENGTH 80
#define NO_CURRENT_STATE ( ( SD_STATE )-1 )

#define SD_CHANNEL_BACKUP 5


// in mmsystem.h the following defines exist...
//#define SND_SYNC              0x0000  /* play synchronously (default) */
//#define SND_ASYNC             0x0001  /* play asynchronously */
//#define SND_NODEFAULT         0x0002  /* don't use default sound */
//#define SND_MEMORY            0x0004  /* lpszSoundName points to a memory file *
  => /
//#define SND_LOOP              0x0008  /* loop the sound until next sndPlaySound
  => */
//#define SND_NOSTOP            0x0010  /* don't stop any currently playing sound
  => */
#define SND_TOGGLEMIC        0x0020

void beep( char* wavefile, int mode=1 ); // SND_ASYNC

int fileNameCmp( const char *f1, const char *f2 );

#define SCP_PROMPT  1


//////////////////////////////////////////////////////////////////////////
// SpeechTask
//////////////////////////////////////////////////////////////////////////

class SpeechTask {
 protected:
    static char versbuf[128];
 public:
    SD_TASK task;
    static void getPar( const char *parName, char* val );
    static void getPar( const char *parName, int16* val );
    static void getPar( const char *parName, int32* val );
    static void getPar( const char *parName, uns32* val );

    static void setPar( const char* parName, char* val );
    static void setPar( const char* parName, int16 val );
    static void setPar( const char* parName, int32 val );
```

DRAGCPP.H 7-22-95 2:12p

```
    static void setPar( const char* parName, uns32 val );

    SpeechTask( char *taskName, void SD_CALLBACK _export (*sdErrHandler)(int, ch
  => ar*) );
    ~SpeechTask();
};


//////////////////////////////////////////////////////////////////////////
// RecogResult
//////////////////////////////////////////////////////////////////////////

class RecogResult : RECOG_RESULT_ENTRY
{
 public:
    int wordCount() const ;
    int getDistance( ) const ;
    int getName( char* textbuf, int sizebuf ) const ;

    SD_VOC vocID( int i=0 ) const ;
    SD_STATE stateID( int i=0 ) const ;
    SD_WORD wordId( int i=0 ) const ;

    SD_STATE jmpTransition( int i=0 ) const;
};


struct ArrayOfRecogResults
{
    RECOG_RESULT_ENTRY recogResults[ MAX_RECOG_CHOICES ];

    RECOGN_STATUS recogStatus;
    TRAIN_STATUS trainStatus;

    SD_USER recogUser;
    SD_VOC recogVoc;
    SD_STATE recogState;

    bool isCont;

    ArrayOfRecogResults() { recogVoc = 0; recogState = 0; recogUser = 0; isCont
  => = 0; recogStatus.nChoices = 0; }
};


//////////////////////////////////////////////////////////////////////////
// Channels
//////////////////////////////////////////////////////////////////////////

class Recognizer;

class UttChannel {
 protected:
    SD_UTT uttUtt;
    int uttIndex, nextIndex;
    char *uttPrompt;
    ArrayOfRecogResults *choices;

 public:

    UttChannel();
    virtual ~UttChannel();

    // utt maintenance        -- Audio Input
    const SD_UTT utt();
    const int index();
```

Page 1 of 9

```
    int setNextIndex( int newValue );

    virtual bool open( SD_CHANNEL_EVENT_HANDLER= 0 );
    virtual void close();

    virtual SD_UTT peekUtt();
    virtual SD_UTT startUtt();
    void endUtt() { }

    virtual void killUtt();       // delete the utt, with the notion of failure
    virtual void nextUtt();       // delete the utt, with the notion of success
    virtual void purge();         // killUtt on buffered utts
=>          // channel
    virtual void flush();         // nextUtt on buffered utts

    virtual bool listen( bool );  // sets a new listen state,
                                      // returns the old listen state (0,1)
    virtual bool isListening(); // returns the current listen state
    virtual bool notifyChannel( SD_CHANNEL, SD_CHANNEL_EVENT );

    // after giveAwayUtt()'s utt=0, prompt = "";
    SD_UTT giveAwayUtt( UttChannel* toMe );
    SD_UTT giveAwayUtt();


    // playing Waves          -- Audio Output
    virtual void beep( char* wavFileName, int mode=1 );


    // prompt maintenance   -- Script Input
    const char* prompt();
    void setPrompt( const char* );
    void seperateDigits( int maxNumDigits );

    virtual bool peekPrompt( char* buf, int bufSize, int promptsAhead=1 );
    virtual bool canBackup( int );
    virtual bool nextPrompt();


    // RecogResult callthroughs
    const RecogResult* result( int ithResult =0 );
    int resultName( char* buf, int sizebuf, int ithResult =0 );
    SD_WORD resultId( int ithResult =0 );
    int resultDistance( int ithResult =1 );
    int resultCount();
    int confidence();

    friend class Recognizer;
    friend class WindowsAdaptChannel;
};


class WindowsLiveChannel : public UttChannel {
 protected:
    SD_CHANNEL mic;

    FILE *promptFile;
    char *promptFilename;

    int flag;

    bool chanIsOn;

 public:
    WindowsLiveChannel( const char* promptFname=0 );
    ~WindowsLiveChannel();
```

```
    bool open( SD_CHANNEL_EVENT_HANDLER );
    void close();

    bool nextPrompt();
    void nextUtt();
    SD_UTT peekUtt();

    bool listen( bool );
    bool isListening();
    bool notifyChannel( SD_CHANNEL, SD_CHANNEL_EVENT );

    void beep( char* wavFileName, int mode = 1 );

};


class WindowsFileChannel : public WindowsLiveChannel
{
 protected:
    SD_CHANNEL uttFile;
    char *uttFilename;

    SD_CHANNEL_EVENT_HANDLER notifyRoutine;

 public:
    WindowsFileChannel(  const char* token, const char* prompt=0 );
    ~WindowsFileChannel();

    bool open( SD_CHANNEL_EVENT_HANDLER );
    void close();

    SD_UTT startUtt();
    SD_UTT peekUtt();
    void nextUtt();

    void killUtt();

    bool listen( bool );
    bool notifyChannel( SD_CHANNEL, SD_CHANNEL_EVENT );
};


#define SIZE_UTT_RING 16

class WindowsCollectChannel : public WindowsFileChannel {

 protected:
    UttChannel uttRing( SIZE_UTT_RING );
    int head, tail, point, advance;

    int incIndex( int i );
    int decIndex( int i );

    int incHead();

    virtual void outputUtt( UttChannel* uCh );      // sends out the utt
                                                    // then kills it

 public:
    WindowsCollectChannel( const char* uttFname, const char* promptFname );
    ~WindowsCollectChannel();

    bool open( SD_CHANNEL_EVENT_HANDLER );
    void close();
    SD_UTT startUtt();
    SD_UTT peekUtt();
    void nextUtt();

    bool canBackup( int numBack );
    bool peekPrompt( char* buf, int bufSize, int promptsAhead =1 );
    bool nextPrompt();

    void killUtt();
    void purge();       // killUtt on buffered utts
  => // channel
    void flush();       // nextUtt on buffered utts

    bool listen( bool );
    bool notifyChannel( SD_CHANNEL, SD_CHANNEL_EVENT );
};


class WindowsAdaptChannel : public WindowsCollectChannel {
 protected:
    void outputUtt( UttChannel* uCh );
    Recognizer *recog;

 public:
    WindowsAdaptChannel( const char* promptFname );
    ~WindowsAdaptChannel();

    bool open( SD_CHANNEL_EVENT_HANDLER );
    void close();

    // DANGER DANGER... we create a temporary recognizer to do buffered
    // adapts... We use the voc and user information in the channel.choice
    // list, the user and voc must still being around, you must not
    // explicitly unloaded them... You are responsible for calling nextUtt()
    // and flush() or killUtt() and purge() when you discard vocs or
    // users... you must clear the buffered utts and do your adapts before
    // you switch them out
};


/////////////////////////////////////////////////////////////////////
// Recognizer Parts
/////////////////////////////////////////////////////////////////////

class VocStateMachine;


class VocStateMachine : public DLink< VocStateMachine > {
     // A slave class of a recognizer...
     // The vocabulary is a user independent part of the database

 protected:
     static Deq< VocStateMachine > vsmDeq;
     static SD_STATE currentState;

     SD_STATE state;
     SD_VOC voc;

     SD_WORD_ITERATOR state0WordIterator;
     SD_STATE_WORD_ITERATOR stateWordIterator;
     SD_STATE state0Iterator;   // -1 none, 0 allWords, nonzer0 normal


     SD_STATE_CHILD_ITERATOR childIterator;

     void loadPhonetics();
```

```
    VocStateMachine();
    ~VocStateMachine();

    SD_VOC open( const char* );
    void save( const char* =0 );
    void close( const char* =0 );

    SD_VOC setVoc( SD_VOC );
    SD_VOC setVoc( const char* vname );
    SD_VOC getVoc();
    int getVocname( char* buf, int sizebuf, SD_VOC vTest=0 );


    SD_STATE setState( SD_STATE hState );
    SD_STATE setState( const char* statename, SD_STATE parent=0 );

    SD_STATE getState( const char* statename, SD_STATE parent=0 );
    SD_STATE getState();

    SD_STATE firstChild( SD_STATE );
    SD_STATE firstChild();
    SD_STATE nextChild();

//    void updateState( SD_WORD );
//    void updateState( const class RecogResult* );

    void listState( long numWrds =0, FILE* fout =0 );
    long wordCount();

    SD_WORD firstWordId( SD_STATE=-1 );
    SD_WORD nextWordId();
    SD_WORD nthWordId( long index );

    SD_WORD wordId( const char *name );

    bool wordIsActive( SD_WORD hWord );

    SD_STATE jmpTransition( SD_WORD hWord );

    size_t wordName( SD_WORD w, char *buf, size_t buflen );
    size_t stateName( SD_STATE w, char *buf, size_t buflen );

    bool wordHasModel( SD_WORD w );

    SD_STATE newState( const char *newStateName );
    void includeState( SD_STATE stateToInclude );
    void deleteState( SD_STATE hState );

    // addWord()'s and buildWord()'s add to the current voc and state
    SD_WORD addWord( SD_WORD );
    SD_WORD addWord( const char* );
    SD_WORD buildWord( const char* spelling, const char* phrase );

    bool deleteWord( SD_WORD ); // currently does not search states
    bool deleteWord( const char* );    // currently does not search states

    bool removeWordFromState( SD_WORD );
    bool removeWordFromState( const char* );

    friend class Recognizer;
};


class UserPhonetics;
class UserPhonetics : public DLink< UserPhonetics >
{
    // Another slave class of a recognizer...
    // This is the user dependent phoneme set

 protected:
    static Deq< UserPhonetics > upDeq;

    static SD_USER activeUser;

    SD_USER user;

    UserPhonetics();
    ~UserPhonetics();

    SD_USER open( const char* );
    void close( const char* =0 );
    void save( const char* =0 );

    SD_USER setUser( SD_USER u );
    SD_USER setUser( const char* );
    SD_USER getUser();

    int getUsername( char* buf, int sizebuf, SD_USER uTest=0 );

    void active();

    friend class Recognizer;
};


class Recognizer
{
 protected:
    UttChannel *channel;
    UserPhonetics userPhonetics;
    //  UserLanguageModel userLanguageModel;
    VocStateMachine vsm;

    RECOGX_PARAMETERS recogPars;
    TRAIN_PARAMETERS trainPars;

    ADAPTLM_PARAMETERS adaptLmPars;
    bool adaptLm;

 public:

    Recognizer( UttChannel* );
    ~Recognizer();

    SD_REJCODE recog( AC< SD_WORD > *passThru =0);
    SD_REJCODE contRecog( int maxWords =0, int minWords =0);

    SD_REJCODE adapt();
    SD_REJCODE contAdapt();

    SD_REJCODE collect();


    void setRecogPars( int16 maxC=MAX_RECOG_CHOICES,
                       int16 rejl=1,              int16 distT=1000,
                       int16 ulm=NO,              int16 comput=5,
```

```
                       SD_CONTEXT far *pCon=0,  char far *pref=0 );

    void setTrainPars( int32 wght =0, int16 tlrnce=3 );

    void setAdaptLmPars( bool adptLm =0,
                         int32 lTmRel =-1000000, int32 sTmRel =-100000,
                         SD_CONTEXT *ctxt=0 );

    void setPars( );


    // Channel call throughs
    UttChannel* getChannel();
    UttChannel* setChannel( UttChannel* newCh );

    const char* prompt();
    void setPrompt( const char* );

    int promptIndex();
    int setNextPromptIndex( int newValue );

    SD_UTT utt();

    bool peekPrompt( char* buf, int bufSize, int promptsAhead=1 );
    bool nextPrompt();

    void seperateDigits( int maxNumDigits );

    bool isListening();
    bool listen( bool );
    bool notifyChannel( SD_CHANNEL ch, SD_CHANNEL_EVENT ev );

    void nextUtt();
    SD_UTT peekUtt();
    void killUtt();

    SD_UTT giveAwayUtt( UttChannel* toMe );
    SD_UTT giveAwayUtt();

    const RecogResult* result( int ithResult=0 );
    SD_WORD resultId( int ithResult=0 );
    int resultName( char* buf, int sizebuf, int ithResult=0 );
    int resultDistance( int ithResult=1 );
    int resultCount();
    int confidence();

    void beep( char* wavFileName, int mode = 1 );

    // VocStateMachine call throughs
    SD_WORD wordId( const char *name );

    SD_VOC setVoc( SD_VOC voc );
    SD_VOC setVoc( const char *name );

    SD_VOC getVoc( );
    int getVocname( char* buf, int sizebuf, SD_VOC vTest=0 );

    void saveVoc( const char* name=0 );

    void closeVoc(const char* name=0);

    SD_STATE setState( SD_STATE hState );
    SD_STATE setState( const char *name, SD_STATE parent=0 );

    SD_STATE getState( const char* statename, SD_STATE parent=0 );
    SD_STATE getState();
    SD_STATE firstChildState( SD_STATE parent );
    SD_STATE firstChildState();
    SD_STATE nextChildState();

//    void updateState( SD_WORD w );
//    void updateState( const char* wordname );
//    void updateState( const RecogResult *res );

    void listState( long numWrds =0, FILE* fout =0 );
    SD_WORD firstWordId( SD_STATE st=-1 ) { return vsm.firstWordId( st ); }
    SD_WORD nextWordId() { return vsm.nextWordId(); }
    SD_WORD nthWordId( long index ) { return vsm.nthWordId( index ); }

    long wordCount();

    size_t wordName( SD_WORD w, char *buf, size_t buflen );
    size_t stateName( SD_STATE hState, char *buf, size_t buflen );

    bool wordHasModel( SD_WORD w );
    bool wordHasModel( const char* name );

    bool wordIsTrained( SD_WORD w );
    bool wordIsTrained( const char * ) ;

    bool wordIsActive( SD_WORD w );
    bool wordIsActive( const char* name );

    SD_STATE impTransition( SD_WORD hWord );
    SD_STATE impTransition( const char* name );

    void Recognizer::deleteState( SD_STATE hState );
    SD_STATE Recognizer::newState( char *newStateName );
    void Recognizer::includeState( SD_STATE hState );

    // addWord()'s and buildWord()'s add to the current voc and state
    SD_WORD addWord( SD_WORD );
    SD_WORD addWord( const char* );
    SD_WORD buildWord( const char* spelling, const char* phrase );

    bool deleteWord( SD_WORD ); // currently does not search states
    bool deleteWord( const char* );     // currently does not search states

    bool removeWordFromState( SD_WORD );
    bool removeWordFromState( const char* );


    // UserPhonetics call throughs
    SD_USER setUser( SD_USER usr );
    SD_USER setUser( const char* name );
    SD_USER getUser() { return userPhonetics.getUser(); }
    int getUsername( char* buf, int sizebuf, SD_USER uTest=0 );

    void saveUser( const char* name=0 );

    void closeUser( const char* name=0 );
};


#if 0
class PhonemeRecognizer : public Recognizer
{
 protected:

 public:
```

DRAGCPP.H 7-22-95 2:12p

```
    PhonemeRecognizer( UttChannel* );
    SD_REJCODE recog();
};
#endif


////////////////////////////////////////////////////////////////////////
////////    Inline Member Functions ///////////////////////////////////
////////////////////////////////////////////////////////////////////////



////////////////////////////////////////////////////////////////////////
// Channels
////////////////////////////////////////////////////////////////////////

inline const SD_UTT UttChannel::utt() { return uttUtt; }

inline const int UttChannel::index() { return uttIndex; }

inline int UttChannel::setNextIndex( int newValue )
{
    return nextIndex = newValue;
}
inline const char* UttChannel::prompt() { return uttPrompt; }


inline SD_WORD UttChannel::resultId( int ithResult )
{
    const RecogResult *res = result( ithResult );

    return( res ? res->wordId() : 0 );
}

inline int UttChannel::resultDistance( int ithResult )
{
    const RecogResult *res = result( ithResult );

    return( res ? res->getDistance() : -1 );
}



inline int UttChannel::resultCount()
{
    return choices->recogStatus.nChoices;
}

inline int UttChannel::confidence()
{
    return choices->recogStatus.confidence;
}


inline int WindowsCollectChannel::incIndex( int i )
{
    return( (i+1) & (SIZE_UTT_RING-1) );
}
inline int WindowsCollectChannel::decIndex( int i )
{
    return( i==0 ? SIZE_UTT_RING-1 : i-1 );
}
```

Page 5 of 9

```
/////////////////////////////////////////////////////////////////////
// VocStateMachine
/////////////////////////////////////////////////////////////////////


inline void VocStateMachine::loadPhonetics() {  // after opening VocStateMachine
  => & UserPhonetics
    assert( voc );    SDWord_Load( voc, 0 /* all ids */ );
}

inline VocStateMachine::VocStateMachine()
{
    voc = 0;    state = NO_CURRENT_STATE;
    stateOfIterator = -1;
    vsmDeq.addFirst( this );
}
inline VocStateMachine::~VocStateMachine()
{
    close(); vsmDeq.remove( this );
}

inline SD_VOC VocStateMachine::setVoc( SD_VOC v )
{
    return( voc = v );
}

inline SD_VOC VocStateMachine::getVoc()
{
    return voc;
}

inline SD_STATE VocStateMachine::setState( SD_STATE sdstate )
{
    return state = sdstate;
}
inline SD_STATE VocStateMachine::setState( const char* statename, SD_STATE paren
  => t )
{
    return setState( getState( statename, parent ) );
}

inline SD_STATE VocStateMachine::getState()
{
    return state;
}

inline SD_STATE VocStateMachine::firstChild()
{
    return( firstChild( getState() ) );
}

inline SD_STATE VocStateMachine::firstChild( SD_STATE parent )
{
    SDState_IterateChildren( getVoc(), parent, &childIterator );

    return SDState_NextChild( &childIterator );
}

inline SD_STATE VocStateMachine::nextChild()
{
    return SDState_NextChild( &childIterator );
}


inline SD_WORD VocStateMachine::wordId( const char *name )
                        DRAGCPP.H  7-22-95  2:12p
```

```
{
    assert( voc && name ); return SDWord_GetHandle( voc, name );
}

inline size_t VocStateMachine::wordName( SD_WORD w, char *buf, size_t buflen )
{
    if( voc && w )
        return SDWord_GetName( voc, w, buf, buflen );
    if( buflen && buf )
        *buf='\0';
    return 0;
}
inline size_t VocStateMachine::stateName( SD_STATE st, char *buf, size_t buflen
  => )
{
    if( voc && st )
        return SDState_GetName( voc, st, buf, buflen );
    if( buflen && buf )
        *buf='\0';
    return 0;
}



/////////////////////////////////////////////////////////////////////
// UserPhonetics
/////////////////////////////////////////////////////////////////////

inline UserPhonetics::UserPhonetics()
{
    user = 0; upDeq.addFirst( this );
}
inline UserPhonetics::~UserPhonetics()
{
    close(); upDeq.remove( this );
}

inline SD_USER UserPhonetics::getUser()
{
    return user;
}
inline void UserPhonetics::active()
{
    if( activeUser!=user ) SDUser_SetCurrent( activeUser = user );
}


/////////////////////////////////////////////////////////////////////
// RecogResult
/////////////////////////////////////////////////////////////////////

inline int RecogResult::wordCount() const
{
    return nWords;
}
inline int RecogResult::getDistance( ) const
{
    return distance;
}

inline SD_VOC RecogResult::vocID( int i ) const
{
    assert( i<wordCount() );    return wordSpec[ i ].hVoc;
}
inline SD_STATE RecogResult::stateID( int i ) const
                        Page 6 of 9
```

```
{
    assert( i<wordCount() );    return wordSpec[ i ].hState;
}
inline SD_WORD RecogResult::wordId( int i ) const
{
    assert( i<wordCount() );    return wordSpec[ i ].hWord;
}



/////////////////////////////////////////////////////////////////////
// Recognizer
/////////////////////////////////////////////////////////////////////

inline void Recognizer::setRecogPars( int16 maxC,
                                      int16 rejT,          int16 distT,
                                      int16 ulm,           int16 comput,
                                      SD_CONTEXT far *pCon, char far *pref )
{
    assert( maxC <= MAX_RECOG_CHOICES );
    recogPars.maxChoices = maxC;
    recogPars.rejThresh = rejT;
    recogPars.distThresh = distT;
    recogPars.useLm = ulm;
    recogPars.computation = comput;
    recogPars.pContext = pCon;
    recogPars.prefix = pref;
    recogPars.pPassThru = 0;
    recogPars.nPassThru = 0;

#ifndef NOMURA
//    SpeechTask::setPar( "use-pelconf", (short )1 );
#endif

    SpeechTask::setPar( "rejection-distance", (short )240 );
}

inline void Recognizer::setTrainPars( int32 wght, int16 tlrnce )
{
    trainPars.weight = wght;
    trainPars.tolerance = tlrnce;
}
inline void Recognizer::setAdaptLmPars( bool adptLm,
                                        int32 lTmRel, int32 sTmRel,
                                        SD_CONTEXT *ctxt )
{
    adaptLm = adptLm;
    adaptLmPars.longTermRelevance = lTmRel;
    adaptLmPars.shortTermRelevance = sTmRel;
    adaptLmPars.pContext = ctxt;
}

inline void Recognizer::setPars( )
{
    setRecogPars(); setTrainPars(); setAdaptLmPars();
}


// Channel call throughs
inline UttChannel* Recognizer::getChannel()
{
    return channel;
}
```

```
{
    assert( newCh );

    UttChannel* oldCh = channel;
    channel = newCh;
    return oldCh;
}

inline bool Recognizer::isListening()
{
    assert( channel );

    return channel->isListening();
}
inline bool Recognizer::listen( bool onOff )
{
    assert( channel );

    return channel->listen( onOff );
}
inline bool Recognizer::notifyChannel( SD_CHANNEL ch, SD_CHANNEL_EVENT ev )
{
    assert( channel );

    return channel->notifyChannel( ch, ev );
}


inline const char* Recognizer::prompt()
{
    assert( channel );  return channel->prompt();
}
inline void Recognizer::setPrompt( const char* p )
{
    assert( channel );  channel->setPrompt( p );
}

inline int Recognizer::promptIndex()
{
    assert( channel );  return channel->index();
}
inline int Recognizer::setNextPromptIndex( int newValue )
{
    return channel->setNextIndex( newValue );
}
inline SD_UTT Recognizer::utt()
{
    assert( channel );  return channel->utt();
}

inline bool Recognizer::peekPrompt( char* buf, int bufSize, int promptsAhead )
{
    return channel->peekPrompt( buf, bufSize, promptsAhead );
}

inline bool Recognizer::nextPrompt()
{
    return channel->nextPrompt();
}

inline const RecogResult* Recogn
```

```
    assert( channel );
    return channel->resultId( ithResult );
}
inline int Recognizer::resultName( char* buf, int sizebuf, int ithResult )
{
    return channel->resultName( buf, sizebuf, ithResult );
}

inline int Recognizer::resultCount()
{
    return channel->resultCount();
}

inline int Recognizer::confidence()
{
    return channel->confidence();
}
inline int Recognizer::resultDistance( int ithResult )
{
    return channel->resultDistance( ithResult );
}

inline void Recognizer::seperateDigits( int maxNumDigits )
{
    channel->seperateDigits( maxNumDigits );
}

inline SD_UTT Recognizer::peekUtt()
{
    assert( channel );    return channel->peekUtt();
}
inline void Recognizer::nextUtt()
{
    assert( channel );  channel->nextUtt();
}
inline void Recognizer::killUtt()
{
    assert( channel );  channel->killUtt();
}
inline SD_UTT Recognizer::giveAwayUtt( UttChannel* toMe )
{
    return channel->giveAwayUtt( toMe );
}
inline SD_UTT Recognizer::giveAwayUtt()
{
    return channel->giveAwayUtt();
}

inline void Recognizer::beep( char* wavFileName, int mode )
{
    channel
    ? channel->beep( wavFileName, mode )
    : ::beep( wavFileName, mode );
}

// VocStateMachine call throughs
inline SD_WORD Recognizer::wordId( const char *name )
{
    return vsm.wordId( name );
}

inline SD_VOC Recognizer::setVoc( SD_VOC voc )
{
    return( vsm.setVoc( voc ) );
}
```

DRAGCPP.H 7-22-95 2:12p

```
inline SD_VOC Recognizer::getVoc( )
{
    return vsm.getVoc();
}
inline int Recognizer::getVocname( char* buf, int sizebuf, SD_VOC vTest )
{
    return vsm.getVocname( buf, sizebuf, vTest );
}

inline SD_STATE Recognizer::setState( SD_STATE hState )
{
    assert( vsm.voc );  return vsm.setState( hState );
}
inline SD_STATE Recognizer::setState( const char *name, SD_STATE parent )
{
    assert( vsm.voc && name); return vsm.setState( name, parent );
}

inline SD_STATE Recognizer::getState()
{
    assert( vsm.voc );  return vsm.getState();
}

inline SD_STATE Recognizer::getState( const char* statename, SD_STATE parent )
{
    assert( vsm.voc );  return vsm.getState( statename, parent );
}

inline SD_STATE Recognizer::firstChildState( SD_STATE parent )
{
    assert( vsm.voc );  return vsm.firstChild( parent );
}
inline SD_STATE Recognizer::firstChildState()
{
    assert( vsm.voc );  return vsm.firstChild();
}
inline SD_STATE Recognizer::nextChildState()
{
    assert( vsm.voc );  return vsm.nextChild();
}

//inline void Recognizer::updateState( SD_WORD w )
//{
//    assert( vsm.voc );        vsm.updateState( w );
//}
//inline void Recognizer::updateState( const char* wordname )
//{
//    updateState( wordId( wordname ) );
//}
//inline void Recognizer::updateState( const RecogResult *res )
//{
//    assert( vsm.voc );        vsm.updateState( res );
//}

inline void Recognizer::listState( long numWrds, FILE* fout )
{
    vsm.listState( numWrds, fout );
}
inline long Recognizer::wordCount()
{
    return vsm.wordCount();
}

inline size_t Recognizer::stateName( SD_STATE st, char *buf, size_t buflen )
{
    return vsm.stateName( st, buf, buflen );
```

Page 8 of 9

```
}
inline size_t Recognizer::wordName( SD_WORD w, char *buf, size_t buflen )
{
    return vsm.wordName( w, buf, buflen );
}
inline bool Recognizer::wordHasModel( SD_WORD w )
{
    return vsm.wordHasModel( w );
}
inline bool Recognizer::wordHasModel( const char* name )
{
    return wordHasModel( wordId( name ) );
}
inline bool Recognizer::wordIsActive( SD_WORD w )
{
    return( vsm.wordIsActive( w ) );
}
inline bool Recognizer::wordIsActive( const char* name )
{
    return wordIsActive( wordId( name ) );
}
inline void Recognizer::saveVoc( const char* name )
{
    vsm.save( name );
}
inline void Recognizer::closeVoc(const char* name)
{
    vsm.close( name );
}
inline SD_STATE Recognizer::jmpTransition( SD_WORD hWord )
{
    return vsm.jmpTransition( hWord );
}
inline SD_STATE Recognizer::jmpTransition( const char* name )
{
    return jmpTransition( wordId( name ) );
}
inline void Recognizer::deleteState( SD_STATE hState )
{
    vsm.deleteState( hState );
}
inline SD_STATE Recognizer::newState( char *newStateName )
{
    return vsm.newState( newStateName );
}
inline void Recognizer::includeState( SD_STATE hState )
{
    vsm.includeState( hState );
}
inline SD_WORD Recognizer::addWord( SD_WORD w )
{
    return vsm.addWord( w );
}
inline SD_WORD Recognizer::addWord( const char* s )
{
    return vsm.addWord( s );
}
inline SD_WORD Recognizer::buildWord( const char* spelling, const char* phrase )
{
    return vsm.buildWord( spelling, phrase );
```

DRAGCPP.H 7-22-95 2:12p

```
}
inline bool Recognizer::deleteWord( SD_WORD w )
{
    return vsm.deleteWord( w );
}
inline bool Recognizer::deleteWord( const char* s )
{
    return vsm.deleteWord( s );
}
inline bool Recognizer::removeWordfromState( SD_WORD w )
{
    return vsm.removeWordfromState( w );
}
inline bool Recognizer::removeWordfromState( const char* s )
{
    return vsm.removeWordfromState( s );
}


// UserPhonetics call throughs
inline SD_USER Recognizer::setUser( SD_USER usr )
{
    return( userPhonetics.setUser( usr ) );
}
inline void Recognizer::saveUser( const char* name )
{
    userPhonetics.save( name );
}
inline void Recognizer::closeUser( const char* name )
{
    userPhonetics.close( name );
}
inline int Recognizer::getUsername( char* buf, int sizebuf, SD_USER uTest )
{
    return userPhonetics.getUsername( buf, sizebuf, uTest );
}


#endif  // dragcpp_h
```

Page 9 of 9

```
/***********************************************************************

FILE: fepext.h

Description:
        External definitions of all routines in the windows driver dll
        that can be called from another application.

   note: windows.h must have already been included.

Copyright (c) 1986 by Dragon Systems, Inc.

Author:         Hal Strauberg     Created: 03-Sep-1991

Revisions (latest first):

Date:           Author:         Change:
-----           -------         -------
13-Dec-93       ZZN             include "npar.h" to define number of parameters
                                NPAR used in structure SIGPKT
9-Nov-92        ZZN             add structure QUEUE_WAVE
                                and declare sendWaveBuf() for dealing with
                                waveform recording
9-Jun-92        HJS             pass userid's when setting parameters
***********************************************************************/
#include        "npar.h"

#ifndef FEPEXT
#define FEPEXT 1

#ifndef PUBLIC
#define PUBLIC extern
#endif

#ifdef  COLLECT_12KHZ
#define WAVEFRAMESIZE   120
#else
#define WAVEFRAMESIZE   110
#endif

#define LINEAR_16BIT    1
#define LINEAR_8BIT     2
#define USE_ANY_DEVICE 0xFFFF
#define SUPER_USER     0xFFF0

/*
   PACKET STRUCTURES
   There are two kinds of packets: Junior and Senior.  Both packet types
   contain a common header structure, followed by differing data sections.

First, the common header:
*/
struct PKTHEADER
   {
        char status;    /* Packet status */
        char ampterm;   /* Amplitude term for the frame */
        char gain;      /* Gain for the frame */
        char random[3]; /* A random number */
        char nsf;       /* Random number shift count */
        char clipdist;  /* Distance from clipping */
   };


/*
Structure of the packets output by FEP Senior.
```

FEPEXT.H 7-22-95 2:08p

```
*/
struct SIGPKT
   {
        struct PKTHEADER header;        /* Common info */
        char jin[NPAR];                 /* The JIN data */
   };


struct _WAVEFRAME
   {
   uns16 huge data[WAVEFRAMESIZE];
   };
typedef struct _WAVEFRAME WAVEFRAME;

struct _QUEUE_WAVE
   {
   WAVEFRAME huge *base;
   uns16 FAR *frmcnt;
   uns16 maxFrames;
   uns16 framesRecorded;
   uns16 firstFrame;
   } ;

typedef struct _QUEUE_WAVE QUEUE_WAVE;

/*
   waveform format, returned by openWaveRecord()
*/

struct _MYWAVEFORMAT {
        WORD sampleRate;
        WORD sampleFormat;
        WORD hardwareWpid;
        WORD begsil;
        WORD endsil;
        int  xbadj;
};
typedef struct _MYWAVEFORMAT    MYWAVEFORMAT;


/*
   Structure of the parameter block argument for FEP_LISTEN.
        WARNING: The module fepsr.c assumes that all of the fields in this
        parameter block are integers.
*/
struct FEPPARAMS
   {
        int maxgain;
        int sampd;      /* Sample period in usec. */
        int frampd;     /* Frame period in usec. */
        int hthr;       /* Speech detect hi thresh */
        int lthr;       /* Speech detect lo thresh */
        int minsp;      /* Min number of speech frames */
        int endsil;     /* End of utter silence frames */
        int begsil;     /* Begin of utter silence frames */
        int minnz;      /* Squelch threshhold */
        int mulnz;      /* Ampterm multiplier for
                                squelched frames */
        int calibgain;  /* Base gain used for computing amplitudes */
   };

/*
   Values returned by fep_status().
        NOTE!  This structure must allign with A_METER, defined in DRAGON.H.
*/
struct FEP_S_STATUS
```

Page 1 of 2

```
   {
        WORD max_level;
        WORD noise_level;
        WORD speech_level;
        BOOL speech;
        BOOL clipping;
        BOOL noise_warning;
   };

/*
   Structure containing hardware-specific calibration parameters returned by
   fep_calib()
*/
struct CALPARAMS
   {
        int maxgain;
        int hithr;      /* Speech detection high threshhold */
        int lothr;      /* Low threshhold */
        int calibgain;  /* Base gain used for computing amplitudes */
   };

typedef struct
  {
  DWORD dVolume ;
  DWORD dOther ;
  WORD wDeviceID ;
  WORD manID ;
  WORD prodID ;
  } VOLSTRUCT ;

typedef struct
   {
   int dynRange ;
   int maxEnergy ;
   int uttEval ;
   } UTTINFO ;


typedef void (FAR *MM_FUNC_CALLBACK)(BOOL micon);
 long FAR PASCAL Fep_SetMicCallback( MM_FUNC_CALLBACK pCallback ) ;
 int FAR PASCAL Fep_SetMicVolume( VOLSTRUCT FAR *lpVol  ) ;
 int FAR PASCAL Fep_GetMicVolume( VOLSTRUCT FAR *lpVol ) ;
 VOID FAR PASCAL FepSetSpecificDevice( WORD wDevice, BOOL useLowPriorty ) ;


 int FAR PASCAL FepGetUttMeasure( UTTINFO FAR *UttInfo ) ;

#ifndef FEPH
/* Data Structures */
typedef struct Registerstag
  {
  unsigned ax; unsigned bx; unsigned cx; unsigned dx;
  unsigned cs; unsigned ds; unsigned es; unsigned si;
  unsigned di; unsigned bp; unsigned sp; unsigned ip;
  unsigned flags ;
  } REGISTERS, * PREGISTERS, NEAR * NREGISTERS, FAR * LPREGISTERS ;

/* procedure declarations - external */
 BOOL FAR PASCAL DevOpen( int FAR *UserID, BOOL bSetParms ) ;
 BOOL FAR PASCAL DevClose( int UserID ) ;
 WORD FAR PASCAL Fep_FlushQue( void ) ;
 WORD FAR PASCAL Fep_BegSil( void ) ;
 WORD FAR PASCAL Fep_LPkt( void ) ;
 WORD FAR PASCAL GetDevError( LPWORD lpwDevError, BOOL bReset ) ;
 int FAR PASCAL DevRead( int UserID, struct SIGPKT FAR *lpSigPkt,
                         WORD wItemsRequested ) ;
```

FEPEXT.H 7-22-95 2:08p

```
 int FAR PASCAL openWaveRecord( int UserID, MTWAVEFORMAT FAR *waveformat);
 int FAR PASCAL closeWaveRecord( int UserID );

 int FAR PASCAL sendWaveBuf(QUEUE_WAVE FAR *que_wave);
 int FAR PASCAL Fep_Play( LPSTR lpbuf, WORD lbuf,
                          LPSTR (FAR *pisr)(WORD,WORD) ) ;
 int FAR PASCAL Fep_Record( LPSTR lpbuf, WORD lbuf,
                          LPSTR (FAR *pisr)(WORD,WORD) ) ;
 int FAR PASCAL Fep_Sample( struct FEPPARAMS FAR *lpparams, LPSTR lpbuff,
                          WORD block_size, WORD num_blocks ) ;
 int FAR PASCAL GetNumUnreadItems( int UserID ) ;
 int FAR PASCAL Fep_Micon( BOOL FAR * ) ;
 int FAR PASCAL Fep_Stop( int UserID ) ;
 int FAR PASCAL Fep_Status( BOOL FAR *, struct FEP_S_STATUS FAR * ) ;
 int FAR PASCAL Fep_Listen( int UserID, struct FEPPARAMS FAR *, BOOL ) ;
 void FAR PASCAL Fep_CalPars( struct CALPARAMS FAR *lpcalpars ) ;
 int FAR PASCAL Fep_Getpar( WORD id, WORD FAR *lpval ) ;
 int FAR PASCAL Fep_Setpar( WORD UserID, WORD id, WORD val ) ;
 void FAR PASCAL err_scode( int code ) ;
 void FAR PASCAL Fep_Defpar( WORD UserID ) ;
 int FAR PASCAL FepGetChannelInfo( LPSTR szInfo, int cSize ) ;

#endif
#endif
```

Page 2 of 2

```
/* $Header:   F:/usr/local/src/lib/vcs/liberr.h_v   1.2   12 Mar 1992 14:46:24
  => DEAN  $ */
/* LIBERR.H

   Header file for ERR module.

   Copyright (c) Dragon Systems,Inc. 1990-1992

   AUTHOR: Dean Sturtevant
   CREATED: 29-Jan-90

   MODIFICATIONS
   $Log:   F:/usr/local/src/lib/vcs/liberr.h_v  $
 *
 *    Rev 1.2   12 Mar 1992 14:46:24   DEAN
 * error handling from new driver
   Error handling scheme used by new driver installed.
*/


#ifndef _ERR_H_
#define _ERR_H_

#include <setjmp.h>

#define ERR_OVERFLOW 0

typedef const char * MODULE;
typedef void (*ERRHANDLERP)(void *);    /* type passed to ErrHandlerP() */
typedef void (*ERRHANDLERI)(int);       /* type passed to ErrHandlerI() */
typedef void (*ERRHANDLER)(void);       /* type passed to ErrHandler() */
typedef void (*ERRHANDLERX)();  /* type passed to ErrRemoveHandler() */

EXTERN jmp_buf *ErrPush(void);      /* use only via macro ErrTrap */
EXTERN void ErrHandler(void (*handler)(void));
EXTERN void ErrHandlerP(void (*handler)(void *), void *p);
EXTERN void ErrHandlerI(void (*handler)(int), int i);
EXTERN void ErrError(MODULE module, int code, const char *msgfmt, ...);
EXTERN void ErrContinue(void);
EXTERN void ErrPop(void);
EXTERN void ErrRemoveAndCall(void (*handler)());
EXTERN void ErrRemoveHandler(void (*handler)());
EXTERN const char *ErrGetMessage(void);
EXTERN MODULE ErrGetModule(void);
EXTERN int ErrGetCode(void);
EXTERN BOOL ErrErrorIs(MODULE module,int code);

#define ErrTrap() setjmp(*ErrPush())

/* routines that can be provided by the user */
void ErrFatal(const char *msg); /* report an error stack problem */
void ErrBottomHandler(void);    /* default error handler at bottom of stack*/


/* thinking about C++ ... */
#define ERR_TRY              if(!ErrTrap()) {
#define ERR_CATCH            ErrPop(); } else

#ifdef DEBUG
#define assert(x) if(!(x)) \
    ErrError(__FILE__,__LINE__,"Assertion \"%s\" failed.",#x);
#else
#define assert(x)
#endif

/* Backwards compatibility - please don't use! */
```

LIBERR.H 7-22-95 2:08p

```
EXTERN void ErrReport(const char *msg, ...);
EXTERN const char *ErrMessage(void);

#endif /* _ERR_H_ */
```

Page 1 of 1

```
/* NPAR.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#define NPAR            16
//#define       COLLECT_12KHZ
```

```
/* ONESHOT.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef oneshot_h
#define oneshot_h


#include "seq.h"


class MemSL : public SLink< MemSL >
{
 public:
};


class OneShotHeap : public Seq< MemSL >
{
 protected:
    char *start, *end;
    unsigned heapChunkSize;

 public:

    OneShotHeap( unsigned sz = 2048 );
    ~OneShotHeap();

    void* carveItUpAndGiveItAway( unsigned len );
    void empty();
};

inline void* operator new( unsigned len, OneShotHeap* heap )
{
    return heap->carveItUpAndGiveItAway( len );
}

inline void* operator new[]( unsigned len, OneShotHeap* heap )
{
    return heap->carveItUpAndGiveItAway( len );
}


inline void* operator new( unsigned len, void* alreadyMallocedMemory )
{
    return alreadyMallocedMemory;
}

inline void* operator new[]( unsigned len, void* alreadyMallocedMemory )
{
    return alreadyMallocedMemory;
}


#endif
```

ONESHOT.H 7-22-95 2:10p

Page 1 of 1

```
/* LIBFILE.H*/
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef _FILE_H_
#define _FILE_H_

/* Modifications
    7-Sep-92 DGS
    typedefs of SEEKMODE and READMODE are now conditional.
    22-Feb-92 DGS
    FileWrite's pbuf argument should be const void *, not void *
    8-Mar-90 DGS
    Added GETS and FGETS
    20-Feb-90 DGS
    FileWrite had been misnamed as FileWrite
*/

#ifndef stdin       /* kludge to test inclusion of stdio.h */
#include <stdio.h>
#endif

EXTERN FILE *   FileOpen(const char *name, const char *mode);
EXTERN void     FileClose(FILE *pf);
EXTERN int      FileGetLine(FILE *pf, char *pbuf, size_t bufsize);
EXTERN void     FileWrite(FILE *pf, const void *pbuf, size_t nbytes);

#ifndef _READMODE_DEFINED_
typedef enum {RM_EOFOK,RM_ATEOF,RM_EOFBAD} READMODE;
#define _READMODE_DEFINED_
#endif
EXTERN size_t   FileRead(FILE *pf, void *addr, size_t nbytes, READMODE rm);
EXTERN int32    FileSize(FILE *pf);

#ifndef _SEEKMODE_DEFINED_
typedef enum {SM_BEGINNING,SM_CURRENT,SM_END} SEEKMODE;
#define _SEEKMODE_DEFINED_
#endif
EXTERN void     FileSeek(FILE *pf, int32 nbytes, SEEKMODE mode);
EXTERN int32    FileGetPosition(FILE *pf);

/* macros for replacing ANSI-standard non-robust routines */
/* NOTE: GETS and FGETS return EOF at end of file, not NULL */
/* NOTE: the argument to GETS MUST be declared as a char[] and
         its length knowable by the compiler */
#define GETS(buf) FileGetLine(stdin,buf,sizeof(buf))
/* NOTE: no restriction on pbuf */
#define FGETS(pbuf,n,fp) FileGetLine(fp,pbuf,n)

#endif /* _FILE_H_ */
```

```
/* HYPO.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef hypo_h
#define hypo_h

#include "dragcpp.h"
#include "stack.h"
#include "oneshot.h"


#define BAD     (1000)


class ZipHypothesis
{
 public:
    bool isNewWord;
    SD_WORD word;
    long value;

    char spelling[ UIT_PROMPT_LENGTH ];
    char phrase[ UIT_PROMPT_LENGTH ];

    ZipHypothesis( Recognizer*, long zip, char* stateName );
    ZipHypothesis( Recognizer*, const char*, const char* );
};

class ZipHypoAC : public AC< ZipHypothesis >
{
 public:
    ZipHypoAC( unsigned initialsize = 5, unsigned incrementsize = 5 )
    : AC< ZipHypothesis >( initialsize, incrementsize ) { }

    void cleanUpNewWords( Recognizer* );
};


////////////////////////////////////////////////////////////////////////


class Hypothesis
{
 protected:
    static int confusionScore[ 100 ];

 public:
    static OneShotHeap heap;

    int wordHistoryArray[ 5 ];

    long currentScore;
    long nextWordIndex;


    Hypothesis() { currentScore = BAD; nextWordIndex = 0; }

    Hypothesis( Hypothesis& h ) {
        currentScore = h.currentScore;
        nextWordIndex = h.nextWordIndex;
        memcpy( wordHistoryArray, h.wordHistoryArray, sizeof( wordHistoryArray )
  *>  );
    }

    void propagate( PriorityQue<Hypothesis>* hpq, int* data, int stateId );
```

HYPO.H 7-22-95 2:12p

```
    long value() {
        return( wordHistoryArray[ 0 ] * 10000L + wordHistoryArray[ 1 ] * 1000L
                + wordHistoryArray[ 2 ] * 100L + wordHistoryArray[ 3 ] * 10L
                + wordHistoryArray[ 4 ] );
    }
};


extern void buildHypothesis( Recognizer*, ZipHypoAC*, char*, char*, int );


#endif // hypo_h
```

Page 1 of 1

```
/* SEQ.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef seq_h
#define seq_h

#include "defs.h"
#include <assert.h>
#include <string.h>


//////////////////////////////////////////////////////////////////////////////
// SLink Template
//////////////////////////////////////////////////////////////////////////////

class SLinkBase
{
 private:
    SLinkBase *nextLink;
 protected:
    void setNext( SLinkBase* list ) { nextLink=list; }
    void addNext( SLinkBase* );

 public:
    SLinkBase() { nextLink = 0; }
    SLinkBase *next() const { return nextLink; }

    friend class SeqBase;
    friend class HashSeqBase;
};


template <class T>
class SLink : public SLinkBase
{
 protected:
    void setNext( T* list ) { SLinkBase::setNext( (SLinkBase*) list ); }
    void addNext( T* list ) { SLinkBase::addNext( (SLinkBase*) list ); }
    // void remove() { SLinkBase::remove(); }

 public:
    T *next() const { return (T*) SLinkBase::next(); }
};


typedef int (*SLinkCompare)( SLinkBase*, SLinkBase* );
//      compare(SLink *a, SLink *b) ==> (a - b)


//////////////////////////////////////////////////////////////////////////////
// Seq Template
//////////////////////////////////////////////////////////////////////////////

class SeqBase
{
 protected:
    SLinkBase *f, *l;

 public:
    SeqBase() { f=l=0; }

    // SLinkBase manipulation routines
    SLinkBase* first() const { return f; }
    SLinkBase* last() const { return l; }

    void addFirst( SLinkBase* );
```

SEQ.H 7-22-95 2:12p

```
    void addLast( SLinkBase* );
    void addNext( SLinkBase* list, SLinkBase* newItem );
    void addPrev( SLinkBase* list, SLinkBase* newItem );

    void remove( SLinkBase* list );
    #if DEBUG
    void removeAll();
    #else
    void removeAll() { f=l=0; }
    #endif

    bool isEmpty() const        { return ( f == 0 ); }

    void sort( SLinkCompare );

               #if DEBUG
    bool contains( SLinkBase* );
    bool sorted( SLinkCompare );
    long count() const;
    ~SeqBase() { removeAll(); }
    bool validateSeq();
               #endif
};


template <class T>
class Seq : public SeqBase {
 public:
    // SLinkBase manipulation routines
    T* first() const { return (T*) SeqBase::first(); }
    T* last() const { return (T*) SeqBase::last(); }

    void addFirst( T* list ) { SeqBase::addFirst( (SLinkBase*) list ); }
    void addLast( T* list ) { SeqBase::addLast( (SLinkBase*) list ); }
    void addNext( T* list, T* newItem ) { SeqBase::addNext( (SLinkBase*) list, (
  => SLinkBase*) newItem ); }
    void addPrev( T* list, T* newItem ) { SeqBase::addPrev( (SLinkBase*) list, (
  => SLinkBase*) newItem ); }

    void remove( T* list ) { SeqBase::remove( (SLinkBase*) list ); }

               #if DEBUG
    bool contains( T* list ) { return SeqBase::contains( (SLinkBase*) list ); }
               #endif
};


//////////////////////////////////////////////////////////////////////////
//  The HashSeq class is ...
//////////////////////////////////////////////////////////////////////////


class HashSeqBase
{
 public:
    // constructors
    HashSeqBase( unsigned maxSize = 8192, unsigned initialSize = 64 );
    // destructor
    ~HashSeqBase();

    int count() const ;

    SLinkBase** first() const ;
    SLinkBase** last() const ;
    SLinkBase** next( SLinkBase** p ) const;
```

Page 1 of 2

```
    SlinkBase** prev( SLinkBase** p ) const;

    SLinkBase* add( SLinkBase* v );
    // adds v only if v does not already exist in the table (uses compare())
    // returns the pointer in the table (v or the element that compare()s)

    SLinkBase* find( const SLinkBase* v );

    SLinkBase* firstCollision( SLinkBase* v );

    SLinkBase* remove( const SLinkBase* v );
    void removeAll();

    virtual int compare( const SLinkBase*, const SLinkBase* ) const = 0;
    // compare returns 0 when there is a match (like strcmp())

    #ifdef DEBUG
    int checkCount() const;
    #endif

 protected:
    unsigned arraySize;
    unsigned arrayUsed;
    unsigned arrayMaxSize;

    long numSLinks;

    SLinkBase** array;

    virtual unsigned keyHash( const SLinkBase* ) const = 0;

    bool inArrayBounds( SLinkBase** pp ) const;

    void growArray();
);

inline HashSeqBase::~HashSeqBase()
(
    delete [] array;
)

inline int HashSeqBase::count() const
(
    #ifdef DEBUG
    assert( checkCount() );
    #endif
    return numSLinks;
)

inline void HashSeqBase::removeAll()
(
    arrayUsed = 0;
    numSLinks = 0;

    memset( array, 0, arraySize * sizeof( void* ) );
)

inline SLinkBase* HashSeqBase::firstCollision( SLinkBase* v )
(
    return array[ keyHash( v ) & (arraySize-1) ];
)
```

SEQ.H 7-22-95 2:12p

```
template <class I>
class HS : public HashSeqBase
(
 public:
    HS( unsigned maxSize = 8192, unsigned initialSize = 64 )
    : HashSeqBase( maxSize, initialSize ) ( )

    I** first() ( return (I** ) HashSeqBase::first(); )
    I** last() ( return (I** ) HashSeqBase::last(); )
    I** next( I** e ) ( return (I** ) HashSeqBase::next( (SLinkBase** )e ); )
    I** prev( I** e ) ( return (I** ) HashSeqBase::prev( (SLinkBase** )e ); )

    I* add( I* e ) ( return (I*) HashSeqBase::add( (SLinkBase*)e ); )

    I* remove( I* e ) ( return (I*) HashSeqBase::remove( (SLinkBase* ) e ); )

    I* find( I* e ) ( return (I*) HashSeqBase::find( (SLinkBase* )e ); )

    I* firstCollision( I* e ) ( return (I*) HashSeqBase::firstCollision( (SLinkB
 => ase* )e ); )

    int compare( const SLinkBase*, const SLinkBase* ) const = 0;

 protected:
    unsigned keyHash( const SLinkBase* ) const = 0;
);

extern unsigned stringHash( const char* s );    // a nice string hash

#endif /* seq_h */
```

Page 2 of 2

```
/* $Header:   F:/work/newd/inc/vcs/sdapi.h_v   1.72   21 Feb 1994 09:37:42   JED
 *>   $ */
/* FILE: sdapi.h

    DRAGON SYSTEMS CONFIDENTIAL

    Copyright (c) Dragon Systems, Inc. 1991-1994

    AUTHOR: Jed Roberts
    CREATED: 09-Jul-91

    DESCRIPTION
    This is the header file for the Dragon Speech Driver Application
    Programming Interface (SDAPI).

    MODIFICATIONS
    $Log:   F:/work/newd/inc/vcs/sdapi.h_v  $

     Revision 6 on Fri Apr 29 20:13:23 1994 by JED
    Driver version 4.13.50
    Added SDUser_NotifyFileRename & SDVoc_NotifyFileRename

     Revision 5 on Mon Apr 11 16:27:54 1994 by JED
    Driver version 4.13.37
    Added SDVoc_GetWeight & SDVoc_SetWeight.
    Removed rename arg to SDWord_CopyToUser.
    Made prefix arg to Recog2 a const.

     Revision 4 on Wed Mar 30 18:55:55 1994 by DEAN
    Driver Version 4.13.33 - cleanup of sdapi*.h
    Buddy = Jed
    Removed include of SDAPIOLD.H
    Conditionally defined SDPAR BOOL, temporarily??
    NOTE to Dragon users - if this breaks your compile, just include
    sdapiold.h explicitly.

     Revision 3 on Tue Mar 29 09:48:37 1994 by ADRIAN
    Driver Version 4.13.30 - Time stamps added.
    Added startTime field in SD_UTT_INFO.
    Buddy==Jed

     Revision 2 on Fri Mar 25 15:05:29 1994 by JED
    Driver version 4.13.29
    Added SDWord_CopyToUser, SDState_Recog2.

     Revision 1 on Thu Mar 17 13:20:16 1994 by DEAN
    Driver Version 4.13.25 - first version under TLIB!
 *
 *      Rev 1.72   21 Feb 1994 09:37:42   JED
 * Driver version 4.13.15
    Added SDWord_HasModel and SDWord_StateRefCount.
 *
 *      Rev 1.71   31 Jan 1994 18:38:52   JED
 * Driver version 4.13.11
    Buddy==Betsy.
    Added SDState_CopyFrom.
 *
 *      Rev 1.70   27 Jan 1994 19:24:24   BETSY
 * Driver Version 4.13.10
 * Macintosh Changes - Really IAN - Buddy = DEAN = GREG
 *
    Macintosh - minor change to allow SDAPI to compile on the Mac.
 *
 *      Rev 1.69   16 Jan 1994 17:32:32   JED
 * Driver version 4.13.4
    Added SDWord_ListMatchesByFrequency.
```

```
 *
 *      Rev 1.68   13 Jan 1994 12:07:52   BETSY
 * Driver Version 4.13.3
    Added SDState_IterateSpecifiedWords and SDState_NextSpecifiedWords.
 *
 *      Rev 1.67   31 Aug 1993 11:19:02   DEAN
 * Driver Version 4.11.127 (GRG's language model stuff)
    Added language model type TEXT_2 and SDVoc_SetLmType1 routine.  (GRG)
 *
 *      Rev 1.66   23 Aug 1993 17:42:18   FRANKM
 * Driver Version 4.10.34
    Added SDWord_BuildFrom
 *
 *      Rev 1.65   11 Aug 1993 21:01:22   BETSY
 * Driver Version 4.11.106
 *
 *      Rev 1.64   28 Jul 1993 12:03:08   BETSY
 * Driver Version 4.11.93
    Added SDUtt_GetLabel, SDUtt_SetLabel and
    SDUtt_Compare.

 *
 *      Rev 1.63   23 Jun 1993 12:02:00   BETSY
 * Driver Version 4.11.64
 *
 *      Rev 1.62   18 Jun 1993 11:38:20   JED
 * Driver Version 4.11.57
    Removed SDWord_Save.
 *
 *      Rev 1.61   09 Jun 1993 14:53:08   BETSY
 * Driver Version 4.11.52
 *
 *      Rev 1.60   20 May 1993 07:10:08   JOELG
 * Driver version 4.11.39
    SDState_ContRecog1 has been modified to take a minimum and maximum
        number of words as parameters to improve recognition.  A
        value of 0 for both of these new parameters will allow an
        unlimited number of words.

 *
 *      Rev 1.59   19 May 1993 16:00:12   JED
 * Driver Version 4.11.35
    passXTable -> pPassXTable for consistency.
    Also, made them far pointers.
 *
 *      Rev 1.58   19 May 1993 09:14:50   DEAN
 * Driver Version 4.11.32
    Made sure all structures had well-defined sizes. In particular,
    removed all enums and introduced BOOL16 and SDPAR_BOOL16.
    Also, on a whim, redefined the STRUC macro.
 *
 *      Rev 1.57   17 May 1993 15:35:18   JOELG
 * Driver version 4.11.28

    The following functions have been renamed.  The original functions
    have been moved to SDAPIOLD.H.

        SDApi_Init          (replaces SDInit)
        SDApi_GetVersion    (replaces SDGetVersion)
        SDUtt_SetWatchDog   (replaced SDSetWatchDog)
        SDUtt_Cancel        (replaces SDQuit)

    The following function has been added as a replacement for the
    original function.  The new function has increased functionality.
    The old function is still available from SDAPIINN.H
```

```
        SDChannel_GetInfo1  (replaces SDChannel_GetInfo)

     In addition, the following functions have been added to support word
     ordering in vocabulary files.

        SDVoc_SetCollationTables
        SDVoc_GetCollationTables

 *
 *     Rev 1.56   13 May 1993 16:22:26   JOELG
 * Driver version 4.11.21
    Cleaned up tabbing by removing all tabs.
 *
 *     Rev 1.55   11 May 1993 14:39:56   JOELG
 * Driver version 4.11.16
    Added SDUser_Open1 and SDVoc_Open1 functions.
 *
 *     Rev 1.54   07 May 1993 15:43:32   JOELG
 * Driver Version 4.11.12
 *
 *     Rev 1.53   06 May 1993 12:12:38   BORIS
 * Driver Version 4.11.08

     The SDAPI.H header file was restructured to prepare it for release
     as part of the Dragon VoiceTools project.  This involved a large
     amount of reformatting and moving of function declarations.

     The following functions are new in this version:

        SDUser_Open1
        SDVoc_Open1
        SDWord_Train1
        SDWord_Learn1
        SDWord_AdaptLm1
        SDPar_Iterate
        SDPar_Next
        SDState_GetInfo1
        SDState_GetWordInfo1
        SDState_Recog1
        SDState_ContRecog1

     The following functions are obselete and have been moved into
     the SDAPIOLD.H file, new use of these functions is discouraged:

        SDChannel_IsClaimed          (replaced with SDChannel_GetInfo)
        SDChannel_IsMicOn            (replaced with SDChannel_GetInfo)
        SDChannel_IsMicSwitchOn      (replaced with SDChannel_GetInfo)
        SDChannel_IsSpeechAvailable  (replaced with SDChannel_GetInfo)
        SDWord_Recog                 (replaced with SDState_Recog1)
        SDState_Save                 (replaced with SDUser_Save)
        SDState_IsWordInState        (replaced with SDState_GetWordInfo1)

     The following functions are not being released as part of VoiceTools
     and have been moved into SDAPIINH.H, these functions are still
     available for inhouse use, although you may find the simplier forms
     developed for the tool kit to be easier to use:

        SDUser_Open                  (replaced with SDUser_Open1)
        SDVoc_Open                   (replaced with SDUser_Voc1)
        SDWord_Train                 (replaced with SDWord_Train1)
        SDWord_Adapt                 (replaced with SDWord_Train1)
        SDWord_Learn                 (replaced with SDWord_Learn1)
        SDWord_AdaptLm               (replaced with SDWord_AdaptLm1)
        SDPar_GetNumPar              (replaced with SDPar_Iterate)
        SDState_GetInfo              (replaced with SDState_GetInfo1)
        SDState_GetWordInfo          (replaced with SDState_GetWordInfo1)
```

```
        SDState_Recog                (replaced with SDState_Recog1)
        SDState_ContRecog            (replaced with SDState_ContRecog1)

        SDState_ContRecogPartial     (not available in VoiceTools)
        SDStMach_New                 (not available in VoiceTools)
        SDStMach_Delete              (not available in VoiceTools)
        SDStMach_Update              (not available in VoiceTools)
        SDStMach_GetCurrentState     (not available in VoiceTools)
        SDCallDriver                 (not available in VoiceTools)
        SDGetFuncId                  (not available in VoiceTools)
        SDGetFuncName                (not available in VoiceTools)
        SDSetFuncCallBack            (not available in VoiceTools)

     The revision history for SDAPI.H prior to this version has been moved
     into SDAPIOLD.H as well.

 **************************************************************************
 */

#ifndef _SDAPI_H_
#define _SDAPI_H_

#ifndef _SID_H_
#define _SID_H_
typedef int             bool,boolean;
typedef short int               BOOL16;
typedef signed char       int8;
typedef short int       int16;
typedef long            int32;
typedef unsigned int      uint,UINT;
typedef unsigned char      uns8,uint8;
typedef unsigned short      uns16,uint16;
typedef unsigned long     uns32,uint32;
#if defined(applec)
#undef   _MACINTOSH_
#define _MACINTOSH_
#endif
#if defined( _HIGHC_ ) || defined(_MACINTOSH_)
#undef   _32BIT_
#define _32BIT_
#endif
#endif // _SID_H_
#ifndef _SIZE_T
#define _SIZE_T
typedef unsigned size_t;
#endif
#ifdef  _32BIT_
#define far
#define _far
#endif

#ifndef FILE_SPEC_DEFINED
#ifdef  _MACINTOSH_
#include <Files.h>
typedef FSSpec FileSpec;
#else
typedef char    FileSpec;
#endif
#define FILE_SPEC_DEFINED
#endif

#ifndef STRUC
#define STRUC(x) typedef struct x x; struct x
    /* structures defined using this macro can be referred to
     * without the 'struct' qualifier */
#endif
```

```
#ifdef __cplusplus
extern "C" {
#endif

#if defined( __HIGHC__ )
#define NOCALLBACKS
#endif

#ifndef _32BIT
#define SD_ENTRYPT _far
#define SD_CALLBACK _far
#else /* for now, just define to be empty string */
#define SD_ENTRYPT
#define SD_CALLBACK
#endif

//------------------------------------------------------------------------
//                          INITIALIZATION
//
//------------------------------------------------------------------------
//

void        SD_ENTRYPT  SDApi_Init( void );
size_t      SD_ENTRYPT  SDApi_GetVersion( char far *pBuf, size_t lBuf );


//------------------------------------------------------------------------
//                          ERROR HANDLING
//
//------------------------------------------------------------------------
//

typedef  void (SD_CALLBACK *SD_ERROR_HANDLER)( int, char far * );

SD_ERROR_HANDLER SD_ENTRYPT SDError_SetHandler( SD_ERROR_HANDLER pHandler );
int              SD_ENTRYPT SDError_GetCode( void );
size_t           SD_ENTRYPT SDError_GetMessage( char far *pBuf, size_t lBuf );


//------------------------------------------------------------------------
//                          TASK MANAGEMENT
//
//  In this implementation, only one task is allowed per application
//
//------------------------------------------------------------------------
//

typedef int16 SD_TASK;

STRUC( SD_TASK_ITERATOR ) {
    char reserved[44];
    };

SD_TASK SD_ENTRYPT SDTask_New( const char far *pName );
void    SD_ENTRYPT SDTask_Close( SD_TASK hTask );
size_t  SD_ENTRYPT SDTask_GetName( SD_TASK hTask, char far *pBuf, size_t lBuf );
SD_TASK SD_ENTRYPT SDTask_GetHandle( const char far *pName );
void    SD_ENTRYPT SDTask_Iterate( SD_TASK_ITERATOR far *pIter );
SD_TASK SD_ENTRYPT SDTask_Next( SD_TASK_ITERATOR far *pIter );
SD_TASK SD_ENTRYPT SDTask_SetCurrent( SD_TASK hTask );
SD_TASK SD_ENTRYPT SDTask_GetCurrent( void );


//------------------------------------------------------------------------
//                          CHANNEL MANAGEMENT
//
// In this implementation, only one device channel is available
//
```

SDAPI.H 7-22-95 2:12p

```
//------------------------------------------------------------------------

typedef uns16 SD_CHANNEL;
typedef uns16 SD_UTT;

STRUC( SD_CHANNEL_ITERATOR ) {
    char reserved[44];
    };

#ifndef _SEEKMODE_DEFINED_
typedef uns16 SEEKMODE;
/* values for SEEKMODE: */
#define SM_BEGINNING    0
#define SM_CURRENT      1
#define SM_END          2
#define _SEEKMODE_DEFINED_
#endif

typedef uns16 SD_CHANNEL_EVENT;
/* values for SD_CHANNEL_EVENT: */
#define         SD_CHANNEL_START            1
#define         SD_CHANNEL_END              2
#define         SD_CHANNEL_LOST_CLAIM       3
#define         SD_CHANNEL_RESTORED_CLAIM   4

typedef uns16 SD_CHANNEL_TYPE;
/* values for SD_CHANNEL_TYPE: */
#define CH_FILE     0
#define CH_DEVICE   1

STRUC( SD_CHANNEL_INFO1 ) {
    BOOL16 isOpen;              /* YES if the channel is open           */
    BOOL16 isClaimed;           /* YES if this channel is claimed       */
    BOOL16 isMicOn;             /* YES if the mic is on for this channel */
    BOOL16 isMicSwitchOn;       /* YES if the phys mic sw is on         */
    BOOL16 isSpeechAvailable; /* YES if speech is available           */
    SD_CHANNEL_TYPE type;   /* CH_FILE or CH_DEVICE                    */
    BOOL16 isUnclaimed;     /* YES if no task has claimed channel      */
    };

/* mode flags for SDChannel_Read: */
#define CH_PEEK      1      /* means peek ahead -- don't consume */
#define CH_WAITSTART 2      /* don't return until an utterance starts */
#define CH_WAITEND   4      /* don't return until the utt ends */

typedef void (SD_CALLBACK *SD_CHANNEL_EVENT_HANDLER)(
                                    SD_CHANNEL, SD_CHANNEL_EVENT );

SD_CHANNEL SD_ENTRYPT SDChannel_NewFile( const FileSpec far *pFileSpec, const ch
  => ar far *pMode );
void    SD_ENTRYPT SDChannel_Open( SD_CHANNEL hChan );
void    SD_ENTRYPT SDChannel_Close( SD_CHANNEL hChan );
SD_UTT  SD_ENTRYPT SDChannel_Read( SD_CHANNEL hChan, int readMode );
void    SD_ENTRYPT SDChannel_Write( SD_CHANNEL hChan, SD_UTT hUtt );
long    SD_ENTRYPT SDChannel_Seek( SD_CHANNEL hChan, long nUtt,
                                   SEEKMODE seekMode );
void    SD_ENTRYPT SDChannel_SetLength( SD_CHANNEL hChan, long nUtt );
void    SD_ENTRYPT SDChannel_Iterate( SD_CHANNEL_ITERATOR far *pCtrl );
SD_CHANNEL  SD_ENTRYPT SDChannel_Next( SD_CHANNEL_ITERATOR far *pCtrl );
size_t  SD_ENTRYPT SDChannel_GetName( SD_CHANNEL hChan, char far *pBuf,
                                      size_t lBuf );
size_t  SD_ENTRYPT SDChannel_GetDeviceName( SD_CHANNEL hChan, char far *pBuf,
                                            size_t lBuf );
size_t  SD_ENTRYPT SDChannel_GetDeviceRev( SD_CHANNEL hChan, char far *pBuf,
                                           size_t lBuf );
size_t  SD_ENTRYPT SDChannel_GetDeviceAddress( SD_CHANNEL hChan, char far *pBuf,
```

Page 3 of 7

```
                                                   size_t lBuf );
bool     SD_ENTRYPT SDChannel_Claim( SD_CHANNEL hChan, bool seize );
void     SD_ENTRYPT SDChannel_Release( SD_CHANNEL hChan );
void     SD_ENTRYPT SDChannel_GetInfo( SD_CHANNEL hChan,
                                       SD_CHANNEL_INFO far *pInfo );
bool     SD_ENTRYPT SDChannel_SetMicOn( SD_CHANNEL hChan, bool purge,
                                       bool micSwitch );
bool     SD_ENTRYPT SDChannel_SetMicOff( SD_CHANNEL hChan );

SD_CHANNEL_EVENT_HANDLER SD_ENTRYPT  SDChannel_SetEventHandler(
                                  SD_CHANNEL hChan, SD_CHANNEL_EVENT event,
                                  SD_CHANNEL_EVENT_HANDLER pHandler );


//------------------------------------------------------------------------
//                          UTTERANCE MANAGEMENT
//
//------------------------------------------------------------------------

typedef int16 SD_REJCODE;
typedef  void (SD_CALLBACK *SD_WATCHDOG)( void );

STRUC( SD_UTT_INFO ) {
    SD_REJCODE rejCode;
    int16 amplitude;
    int16 loudSoft;
    int32 duration;       /* in milliseconds */
    int32 startTime;      /* in milliseconds since midnight */
    };

STRUC( SD_UTT_ITERATOR ) {
    char reserved[44];
    };

void     SD_ENTRYPT SDUtt_Collect( SD_UTT hUtt, SD_UTT_INFO far *pInfo );
void     SD_ENTRYPT SDUtt_GetInfo( SD_UTT hUtt, SD_UTT_INFO far *pInfo );
void     SD_ENTRYPT SDUtt_Delete( SD_UTT hUtt );
void     SD_ENTRYPT SDUtt_Iterate( SD_UTT_ITERATOR far *pCtrl );
SD_UTT   SD_ENTRYPT SDUtt_Next( SD_UTT_ITERATOR far *pCtrl );

SD_WATCHDOG  SD_ENTRYPT  SDUtt_SetWatchDog( SD_WATCHDOG pHandler );
void         SD_ENTRYPT  SDUtt_Cancel( void );

size_t SD_ENTRYPT SDUtt_GetLabel(SD_UTT hUtt, char far *pBuf, size_t lBuf);
void SD_ENTRYPT SDUtt_SetLabel(SD_UTT hUtt, const char far *pUttLabel );
int SD_ENTRYPT SDUtt_Compare(SD_UTT hUtt1, SD_UTT hUtt2 );

//------------------------------------------------------------------------
//                          USER MANAGEMENT
//
//------------------------------------------------------------------------

typedef int16 SD_USER;

STRUC( SD_ENV_ITERATOR ) {
    char reserved[8];
    };

STRUC( SD_USER_ITERATOR ) {
    char reserved[44];
    };

STRUC( SD_USER_INFO ) {
    BOOL16 isModified;    /* the user has been modified       */
    int16  nEnv;            /* number of environment variables  */
    };
```

```
SD_USER SD_ENTRYPT SDUser_OpenI( const FileSpec far *pFileSpec );
SD_USER SD_ENTRYPT SDUser_New( void );
void     SD_ENTRYPT SDUser_Save( SD_USER hUser );
void     SD_ENTRYPT SDUser_SaveAs( SD_USER hUser, const FileSpec far *pFileSpec )
  *> ;
void     SD_ENTRYPT SDUser_Close( SD_USER hUser );
void     SD_ENTRYPT SDUser_GetInfo( SD_USER hUser, SD_USER_INFO far *pInfo );
size_t   SD_ENTRYPT SDUser_GetFileName( SD_USER hUser, FileSpec far *pBuf,
                                        size_t lBuf );
void     SD_ENTRYPT SDUser_SetFileName( SD_USER hUser,
                                        const FileSpec far *pFileSpec );
void     SD_ENTRYPT SDUser_NotifyFileRename( SD_USER hUser,
                                        const FileSpec far *pNewFileSpec );
void     SD_ENTRYPT SDUser_Iterate( SD_USER_ITERATOR far *pIter );
SD_USER SD_ENTRYPT SDUser_Next( SD_USER_ITERATOR far *pIter );
SD_USER SD_ENTRYPT SDUser_SetCurrent( SD_USER hUser );
SD_USER SD_ENTRYPT SDUser_GetCurrent( void );
size_t   SD_ENTRYPT SDUser_GetEnv( SD_USER hUser, const char far *pName,
                                  void far *pBuf, size_t lBuf );
void     SD_ENTRYPT SDUser_SetEnv( SD_USER hUser, const char far *pName,
                                  const void far *pBuf, size_t lBuf );
void     SD_ENTRYPT SDUser_IterateEnv( SD_USER hUser,
                                      SD_ENV_ITERATOR far *pIter );
size_t   SD_ENTRYPT SDUser_NextEnv( char far *pBuf, size_t lBuf,
                                   SD_ENV_ITERATOR far *pIter );
void     SD_ENTRYPT SDUser_ForceResident( SD_USER hUser );
bool     SD_ENTRYPT SDUser_DeleteWord( SD_USER hUser, const char far *pName );


//------------------------------------------------------------------------
//                          VOCABULARY MANAGEMENT
//
//------------------------------------------------------------------------

typedef uns16 SD_VOC;

typedef uns16 SD_LM_TYPE;
/* values for SD_LM_TYPE: */
#define COMMAND_AND_CONTROL 0
#define TEXT_1              1
#define TEXT_2                          3

STRUC( SD_VOC_INFO ) {
    int32       nWords;     /* current number of words in the vocabulary */
    int32       nResident;  /* number of resident words                  */
    BOOL16      isModified; /* the vocab has been modified               */
    SD_LM_TYPE  lmType;     /* default language model                    */
    int16       nEnv;       /* number of environment variables           */
    };

STRUC( SD_VOC_ITERATOR ) {
    char reserved[44];
    };

STRUC( SD_COLLATION_TABLE ) {
    uns8 sequence[256];
    };

SD_VOC   SD_ENTRYPT SDVoc_New( void );
SD_VOC   SD_ENTRYPT SDVoc_OpenI( const FileSpec far *pVocFileSpec,
                                const char far *pMode );
void     SD_ENTRYPT SDVoc_Save( SD_VOC hVoc );
void     SD_ENTRYPT SDVoc_SaveAs( SD_VOC hVoc, const FileSpec far *pFileSpec );
void     SD_ENTRYPT SDVoc_Close( SD_VOC hVoc );
void     SD_ENTRYPT SDVoc_GetInfo( SD_VOC hVoc, SD_VOC_INFO far *pInfo );
```

```
size_t  SD_ENTRYPT SDVoc_GetFileName( SD_VOC hVoc, FileSpec far *pBuf,
                                       size_t lBuf );
void    SD_ENTRYPT SDVoc_SetFileName( SD_VOC hVoc, const FileSpec far *pFileSpec
  »  );
void    SD_ENTRYPT SDVoc_NotifyFileRename( SD_VOC hVoc,
                                          const FileSpec far *pNewFileSpec );
void    SD_ENTRYPT SDVoc_SetLmType( SD_VOC hVoc, SD_LM_TYPE lmType );
void    SD_ENTRYPT SDVoc_SetLmType1( SD_VOC hTargetVoc, SD_VOC hSourceVoc,
                                                                        SD_LM_T
  » YPE lmType );
uns32   SD_ENTRYPT SDVoc_GetWeight( SD_VOC hVoc );
void    SD_ENTRYPT SDVoc_SetWeight( SD_VOC hVoc, uns32 weight );
void    SD_ENTRYPT SDVoc_Iterate( SD_VOC_ITERATOR far *pIter );
SD_VOC  SD_ENTRYPT SDVoc_Next( SD_VOC_ITERATOR far *pIter );
size_t  SD_ENTRYPT SDVoc_GetEnv( SD_VOC hVoc, const char far *pName,
                                  void far *pBuf, size_t lBuf );
void    SD_ENTRYPT SDVoc_SetEnv( SD_VOC hVoc, const char far *pName,
                                  const void far *pBuf, size_t lBuf );
void    SD_ENTRYPT SDVoc_IterateEnv( SD_VOC hVoc, SD_ENV_ITERATOR far *pIter );
size_t  SD_ENTRYPT SDVoc_NextEnv( char far *pBuf, size_t lBuf,
                                   SD_ENV_ITERATOR far *pIter );
void    SD_ENTRYPT SDVoc_SetCollationTables( SD_COLLATION_TABLE far *pPass1Table
  » ,
                                             SD_COLLATION_TABLE far *pPass2Table
  »  );
void    SD_ENTRYPT SDVoc_GetCollationTables( SD_COLLATION_TABLE far *pPass1Table
  » ,
                                             SD_COLLATION_TABLE far *pPass2Table
  »  );

//--------------------------------------------------------------------------
//                     VOCABULARY WORD MANAGEMENT
//
//--------------------------------------------------------------------------

#define SD_ALL_IDS 0

STRUC( SD_WORD_INFO ) (
    BOOL16 hasModel;         /* has an acoustic model and thus can be */
                           /*   recognized (if loaded and active)   */
    BOOL16 isLoaded;         /* model is resident in memory  */
    BOOL16 isTrained;        /* user has trained this word   */
    BOOL16 isModified;       /* word has been modified       */
    uns16  stateRefCount;    /* number of states that reference the word */
    );

STRUC( SD_WORD_ITERATOR ) (
    char reserved[10];
    );

typedef uns32 SD_WORD;

SD_WORD SD_ENTRYPT SDWord_New( SD_VOC hVoc, const char far *pName );
void    SD_ENTRYPT SDWord_Load( SD_VOC hVoc, SD_WORD hWord );
void    SD_ENTRYPT SDWord_Unload( SD_VOC hVoc, SD_WORD hWord );
void    SD_ENTRYPT SDWord_Delete( SD_VOC hVoc, SD_WORD hWord );
SD_WORD SD_ENTRYPT SDWord_GetHandle( SD_VOC hVoc, const char far *pName );
size_t  SD_ENTRYPT SDWord_GetName( SD_VOC hVoc, SD_WORD hWord,
                                    char far *pBuf, size_t lBuf );
void    SD_ENTRYPT SDWord_SetName( SD_VOC hVoc, SD_WORD hWord,
                                    const char far *pNewName );
void    SD_ENTRYPT SDWord_GetInfo( SD_VOC hVoc, SD_WORD hWord,
                                    SD_WORD_INFO far *pInfo );
SD_WORD SD_ENTRYPT SDWord_CopyFrom( SD_VOC hTargetVoc, SD_VOC hSourceVoc,
                                     SD_WORD hSourceWord );
SD_WORD SD_ENTRYPT SDWord_BuildFrom( SD_VOC hTargetVoc,
```

SDAPI.H 7-22-95 2:12p

```
                                      const char far *pName,
                                      int nSourceVocs,
                                      const SD_VOC far *phSourceVocs,
                                      int nSourceWords,
                                      const char far *pSourceWords,
                                      int nUtts, const SD_UTT far *pUtts );
void    SD_ENTRYPT SDWord_CopyToUser( SD_USER hUserTarget,
                                       SD_VOC hVoc,
                                       SD_WORD hWord,
                                       const char far *pNewName);
uns32   SD_ENTRYPT SDWord_Lookup( SD_VOC hVoc, const char far *pName );
uns32   SD_ENTRYPT SDWord_List( SD_VOC hVoc, uns32 index,
                                 SD_WORD far *pBuf, uns32 nBuf );
uns32 SD_ENTRYPT SDWord_ListMatchesByFrequency( SD_VOC hVoc,
        const char far *prefix,
        SD_WORD *pBuf, uns32 nBuf,
        bool caseSensitive,
        long (SD_CALLBACK *fBreak)(void));
void    SD_ENTRYPT SDWord_Iterate( SD_VOC hVoc, SD_WORD_ITERATOR far *pIter );
SD_WORD SD_ENTRYPT SDWord_Next( SD_WORD_ITERATOR far *pIter );
bool    SDWord_HasModel(SD_VOC hVoc, SD_WORD hWord);
uns16   SDWord_StateRefCount(SD_VOC hVoc, SD_WORD hWord);

//--------------------------------------------------------------------------
//                        STATE MANAGEMENT
//
//--------------------------------------------------------------------------

typedef uns32 SD_STATE;

STRUC( SD_STATE_INFO1 ) (
    int32        nWords;       /* current number of words in the state    */
    int32        nActive;      /* number of active words                  */
    SD_STATE     hParent;      /* parent state or 0                       */
    int16        nChildStates; /* number of child states                  */
    int16        nSubStates;   /* number of included substates            */
    int16        nEnv;         /* number of environment variables         */
    int16        wordOrder;    /* flags indicating the word orderings avail */
    );

STRUC( SD_STATE_CHILD_ITERATOR ) (
    char reserved[12];
    );

/* for use as hParent parameter: */
#define SD_NO_PARENT 0

SD_STATE SD_ENTRYPT SDState_New( SD_VOC hVoc, SD_STATE hParent );
void     SD_ENTRYPT SDState_Delete( SD_VOC hVoc, SD_STATE hState );
SD_STATE SD_ENTRYPT SDState_CopyFrom( SD_VOC hVocTarget,
                                       SD_STATE hParentTarget,
                                       SD_VOC hVocSource,
                                       SD_STATE hStateSource,
                                       bool copyChildren,
                                       bool renameTarget,
                                       const char far *newName);

size_t   SD_ENTRYPT SDState_GetName( SD_VOC hVoc, SD_STATE hState,
                                      char far *pBuf, size_t lBuf );
void     SD_ENTRYPT SDState_SetName( SD_VOC hVoc, SD_STATE hState,
                                      const char far *pStateName );
SD_STATE SD_ENTRYPT SDState_GetHandle( SD_VOC hVoc, const char far *pStateName,
                                        SD_STATE hParent );
void     SD_ENTRYPT SDState_GetInfo1( SD_VOC hVoc, SD_STATE hState,
                                       SD_STATE_INFO1 far *pInfo );
size_t   SD_ENTRYPT SDState_GetEnv( SD_VOC hVoc, SD_STATE hState,
```

Page 5 of 7

```
                                     const char far *pName,
                                     void far *pBuf, size_t lBuf );
void     SD_ENTRYPT SDState_SetEnv( SD_VOC hVoc, SD_STATE hState,
                                     const char far *pName,
                                     const void far *pBuf, size_t lBuf );
void     SD_ENTRYPT SDState_IterateEnv( SD_VOC hVoc, SD_STATE hState,
                                     SD_ENV_ITERATOR far *pIter );
size_t     SD_ENTRYPT SDState_NextEnv( char far *pBuf, size_t lBuf,
                                     SD_ENV_ITERATOR far *pIter );
SD_STATE SD_ENTRYPT SDState_GetParent( SD_VOC hVoc, SD_STATE hState );
void     SD_ENTRYPT SDState_SetParent( SD_VOC hVoc, SD_STATE hState,
                                     SD_STATE hParent );
void     SD_ENTRYPT SDState_IterateChildren( SD_VOC hVoc, SD_STATE hState,
                                     SD_STATE_CHILD_ITERATOR far *pIter );
SD_STATE SD_ENTRYPT SDState_NextChild( SD_STATE_CHILD_ITERATOR far *pIter );


/* -- State Word Management ........................... */

STRUC( SD_STATE_WORD_INFO1 ) {
    BOOL16 isInState;
    BOOL16 isActive;
    int16 nEnv;
    };

STRUC( SD_STATE_WORD_ITERATOR ) {
    char reserved[12];
    };

STRUC( SD_STATE_SPECWORD_ITERATOR ) {
    char reserved[20];
    };

#define SD_ORDER_ALPHA 1
#define SD_ORDER_USER  2

#define SD_WORD_TRAINED 1
#define SD_WORD_UNTRAINED 2
#define SD_WORD_HASMODEL 3
#define SD_WORD_NOMODEL 4

void     SD_ENTRYPT SDState_AddWord( SD_VOC hVoc, SD_STATE hState,
                                     SD_WORD hWord );
void     SD_ENTRYPT SDState_DeleteWord( SD_VOC hVoc, SD_STATE hState,
                                     SD_WORD hWord );
void     SD_ENTRYPT SDState_Load( SD_VOC hVoc, SD_STATE hState );
void     SD_ENTRYPT SDState_Unload( SD_VOC hVoc, SD_STATE hState );
void     SD_ENTRYPT SDState_GetWordInfo1( SD_VOC hVoc, SD_STATE hState,
                                     SD_WORD hWord,
                                     SD_STATE_WORD_INFO1 far *pInfo );
bool     SD_ENTRYPT SDState_SetWordActive( SD_VOC hVoc, SD_STATE hState,
                                     SD_WORD hWord, bool Active );
size_t   SD_ENTRYPT SDState_GetWordEnv( SD_VOC hVoc, SD_STATE hState,
                                     SD_WORD hWord,
                                     const char far *pName,
                                     void far *pBuf, size_t lBuf );
void     SD_ENTRYPT SDState_SetWordEnv( SD_VOC hVoc, SD_STATE hState,
                                     SD_WORD hWord,
                                     const char far *pName,
                                     const void far *pBuf, size_t lBuf );
void     SD_ENTRYPT SDState_IterateWordEnv( SD_VOC hVoc, SD_STATE hState,
                                     SD_WORD hWord,
                                     SD_ENV_ITERATOR far *pIter );
size_t   SD_ENTRYPT SDState_NextWordEnv( char far *pBuf, size_t lBuf,
                                     SD_ENV_ITERATOR far *pIter );
void     SD_ENTRYPT SDState_IterateWords( SD_VOC hVoc, SD_STATE hState,
```

SDAPI.H 7-22-95 2:12p

```
                                     SD_STATE_WORD_ITERATOR far *pIter );
SD_WORD  SD_ENTRYPT SDState_NextWord( SD_STATE_WORD_ITERATOR far *pIter );
void     SD_ENTRYPT SDState_SetWordOrder( SD_VOC hVoc,
                                     SD_STATE hState, int order );
uns32    SD_ENTRYPT SDState_LookupWord( SD_VOC hVoc,
                                     SD_STATE hState, int order,
                                     const char far *pName );
uns32    SD_ENTRYPT SDState_ListWords( SD_VOC hVoc,
                                     SD_STATE hState, int order,
                                     uns32 index,
                                     SD_WORD far *pBuf, uns32 nBuf );
void     SD_ENTRYPT SDState_SetWordIndex( SD_VOC hVoc, SD_STATE hState,
                                     SD_WORD hWord, uns32 index );
void     SD_ENTRYPT SDState_IterateSpecifiedWords(SD_VOC hVoc, SD_STATE hState,
                              int order, int restrict,
                              SD_STATE_SPECWORD_ITERATOR far *pIter);
uns32    SD_ENTRYPT SDState_NextSpecifiedWords( SD_WORD far *pBuf, uns32 nBuf,
                              SD_STATE_SPECWORD_ITERATOR far *pIter);

void     SD_ENTRYPT SDState_AddWordList(SD_VOC hVoc, SD_STATE hState,
              SD_WORD far *pBuf, uns32 nBuf );


/*....... State State Management .........................*/

STRUC( SD_STATE_STATE_ITERATOR ) {
    char reserved[12];
    };

void     SD_ENTRYPT SDState_AddState( SD_VOC hVoc, SD_STATE hState,
                                     SD_STATE hSubState, uint index );
void     SD_ENTRYPT SDState_DeleteState( SD_VOC hVoc, SD_STATE hState,
                                     SD_STATE hSubState );
uint     SD_ENTRYPT SDState_GetStateIndex( SD_VOC hVoc, SD_STATE hState,
                                     SD_STATE hSubState );
size_t   SD_ENTRYPT SDState_GetStateEnv( SD_VOC hVoc, SD_STATE hState,
                                     SD_STATE hSubState,
                                     const char far *pName,
                                     void far *pBuf, size_t lBuf );
void     SD_ENTRYPT SDState_SetStateEnv( SD_VOC hVoc, SD_STATE hState,
                                     SD_STATE hSubState,
                                     const char far *pName,
                                     const void far *pBuf, size_t lBuf );
void     SD_ENTRYPT SDState_IterateStateEnv( SD_VOC hVoc, SD_STATE hState,
                                     SD_STATE hSubState,
                                     SD_ENV_ITERATOR far *pIter );
size_t   SD_ENTRYPT SDState_NextStateEnv( char far *pBuf, size_t lBuf,
                                     SD_ENV_ITERATOR far *pIter );
void     SD_ENTRYPT SDState_IterateStates( SD_VOC hVoc, SD_STATE hState,
                                     SD_STATE_STATE_ITERATOR far *pIter );
SD_STATE SD_ENTRYPT SDState_NextState( SD_STATE_STATE_ITERATOR far *pIter );

/*....... Recognition from States  .......................*/

STRUC( SD_WORD_SPEC ) {
    SD_WORD   hWord;
    SD_STATE  hState;
    SD_VOC    hVoc;
    };

STRUC( SD_CONTEXT ) {
    uns16         nContext;
    SD_WORD_SPEC context[1];  /* actual size is nContext */
    };

/* The recog results buffer will contain a consecutive sequence of */
```

Page 6 of 7

```
/* nChoices copies of RECOG_RESULT_ENTRYs. */
STRUC( RECOG_RESULT_ENTRY ) {
    int16         nWords;
    int16         distance;
    SD_WORD_SPEC wordSpec[1];    /* Actual number of entries is nWords */
    };

STRUC( RECOG_STATUS ) {
    SD_REJCODE rejCode;
    int16      confidence;
    int16      nChoices;
    };

int  SD_ENTRYPT SDState_Recog1( SD_VOC hVoc, SD_STATE hState,
                               SD_UTT hUtt, SD_CONTEXT far *pContext,
                               char far *prefix,
                               RECOG_RESULT_ENTRY far *pResults,
                               size_t lResults,
                               RECOG_STATUS far *pRecStatus );

int  SD_ENTRYPT SDState_ContRecog1( SD_VOC hVoc, SD_STATE hState,
                                   SD_UTT hUtt, SD_CONTEXT far *pContext,
                                   int minWords, int maxWords,
                                   RECOG_RESULT_ENTRY far *pResults,
                                   size_t lResults,
                                   RECOG_STATUS far *pRecStatus );

STRUC(SD_STATE_SPEC) {
        SD_VOC hVoc;
        SD_STATE hState;
};

int SD_ENTRYPT SDState_Recog2(int nStates,
        SD_STATE_SPEC far *pStates,
        SD_UTT hUtt,
        SD_CONTEXT far *pContext,
        const char far *prefix,
        RECOG_RESULT_ENTRY far *pResults,
        size_t lResults,
        RECOG_STATUS far *pRecStatus);


//-----------------------------------------------------------------------
//                           ADAPTATION
//
//-----------------------------------------------------------------------

STRUC( TRAIN_STATUS ) {
    int16 nGood;
    int16 nBad;
    };

void SD_ENTRYPT SDWord_Learn1( SD_VOC hVoc, int nWords, const char far *pName,
                              SD_UTT hUtt, RECOG_RESULT_ENTRY far *pResults,
                              SD_CONTEXT far *pContext,
                              char far *prefix,
                              RECOG_STATUS far *pRecStatus );

void SD_ENTRYPT SDWord_Train1( SD_VOC hVoc, int nWords, const char far *pName,
                              int nUtt, SD_UTT far *pUtt,
                              TRAIN_STATUS far *pStatus,
                              SD_REJCODE far *pRejCodes );

void SD_ENTRYPT SDWord_AdaptIm1( SD_VOC hVoc, SD_WORD hWord,
                                SD_CONTEXT far *pContext );
```

SDAPI.H 7-22-95 2:12p

```
//-----------------------------------------------------------------------
//                           PARAMETER MANAGEMENT
//
//-----------------------------------------------------------------------

typedef uns16 SD_PAR;

#define SDPAR_INT16      0
#define SDPAR_STRING     1
#define SDPAR_OTHER      2
#define SDPAR_UNUSED     3
#define SDPAR_BOOL16     4
#define SDPAR_UNS16      5
#define SDPAR_INT32      6
#define SDPAR_UNS32      7
#define SDPAR_VOID       8
typedef uns16 SDPAR_TYPE;

#ifndef SDPAR_BOOL
 #define SDPAR_BOOL      4          /* for backwards compatibility */
#endif

STRUC( SD_PAR_ITERATOR ) {
    char reserved[4];
    };

SD_PAR     SD_ENTRYPT  SDPar_GetHandle( const char far *pName );
size_t     SD_ENTRYPT  SDPar_GetName( SD_PAR hPar, char far *pBuf,
                                      size_t lBuf );
SDPAR_TYPE SD_ENTRYPT  SDPar_GetType( SD_PAR hPar );
void       SD_ENTRYPT  SDPar_GetValue( SD_PAR hPar, void far *pVal,
                                       size_t lVal );
void       SD_ENTRYPT  SDPar_SetValue( SD_PAR hPar, void far *pVal,
                                       size_t lVal );
void       SD_ENTRYPT  SDPar_Iterate( SD_PAR_ITERATOR far *pIter );
SD_PAR     SD_ENTRYPT  SDPar_Next( SD_PAR_ITERATOR far *pIter );

#ifdef __cplusplus
}   /* extern "C" */
#endif

#endif /* _SDAPI_H_ */
```

Page 7 of 7

```
/* $Header:   f:/work/neud/inc/vcs/sdapiinh.h_v   1.14   04 Feb 1994 13:40:36
  => JED  $ */
/*  SDAPIINH.H

    Header file for INHOUSE SDAPI functions.

    Copyright (c) Dragon Systems,Inc. 1992-1994

    AUTHOR: Dean Sturtevant
    CREATED: 27-Jul-92

    DESCRIPTION
    This header file serves the purposes of SDAPI.H for
    Dragon inhouse SDAPI functions.

    MODIFICATIONS
    $Log:   f:/work/neud/inc/vcs/sdapiinh.h_v  $

     Revision 4 on Thu Apr 14 16:34:42 1994 by DEAN
    Driver Version 4.13.38
    Did some beautification.
    SD_STATE_MACHINE was a void far *, now it's an uns32.
    State Machine modifications:
         - disabled SDStMach_GetCurrentState
         - new routine SDStMach_Parse
         - new transition opcode CALL_GOTO.

     Revision 2 on Thu Apr 07 18:44:37 1994 by JED
    Driver version 4.13.35
    Changed args to SDLm_Reorder.

     Revision 2 on Wed Mar 30 18:55:55 1994 by DEAN
    Driver Version 4.13.33 - cleanup of sdapi*.h
    More cleanup:
    - SDARG structures moved to sdapix.h
    - bool16->BOOL16
    - removed include of sdapix.h
    - more use of STRUC macro
    - no longer define _INHOUSE_SDAPI_FUNCS_

     Revision 4 on Tue Mar 29 14:23:31 1994 by DEAN
    Driver Version 4.13.31 - cleanup
    Buddy = Adrian
    - Changed 'del' argument to SDStMach_Update to 'bool' from 'int16'
    - Redeclared SDWord_SetLmlCount to return void.
    - Removed a PUBLIC from in front of SDLm_Reorder
    - Cleaned up indentation

     Revision 3 on Tue Mar 29 09:48:37 1994 by ADRIAN
    Driver Version 4.13.30 - Time stamps added.
    Added SDUtt_Get1 to support the time_stamp, and left
    SDUtt_Get to provide backward compatibility.

     Revision 2 on Fri Mar 25 15:05:38 1994 by JED
    Driver version 4.13.29
    Added SDLm_Reorder.

     Revision 1 on Thu Mar 17 13:20:16 1994 by DEAN
    Driver Version 4.13.25 - first version under TLIB!
 *
 *     Rev 1.14   04 Feb 1994 13:40:36   JED
 * Driver version 4.13.12
   Added SDWord_SetLmlCount.
 *
 *     Rev 1.13   16 Jan 1994 17:30:32   JED
 * Driver version 4.13.4
```

```
 *
 *     Rev 1.12   01 Sep 1993 20:06:14   DEAN
 * Driver Version 4.11.131
   Added SDLm_DecayAdjust and SDLm_GetCats  (GRG)
 *
 *     Rev 1.11   26 Aug 1993 13:06:12   JOSHUA
 * Driver version 4.11.115
 * Added SDXUtt_MakeRandom, and related functions, like generate_whole_token
 *
 *     Rev 1.10   17 May 1993 15:36:32   JOELG
 * Driver version 4.11.28
   Moved SDChannel_GetInfo in from SDAPI.H
 *
 *     Rev 1.9   11 May 1993 14:40:24   JOELG
 * Driver version 4.11.16 -- added SDUser_Open and SDVoc_Open
 *
 *     Rev 1.8   06 May 1993 12:07:38   JOELG
 * Driver Version 4.11.08
        Added definitions for the functions which were moved out of SDAPI.H
        because we do not want to expose them in the toolkit.  See SDAPI.H
        for more details.
 *
 *     Rev 1.7   24 Feb 1993 15:18:18   JOSHUA
 * Driver version 4.10.36
 * Phoneme recognizer and minor bug fixes.
 *
 *

   Added SDState_PhonRecog.
 *
 *     Rev 1.6   08 Nov 1992 15:07:14   JED
 * Driver Version 4.04.29
   Added SDWord_GetLmlCount.
 *
 *     Rev 1.5   02 Nov 1992 10:19:12   JED
 * Driver Version 4.04.24
   Added SDVoc_GetExtendedInfo & SDState_GetExtendedInfo.
 *
 *     Rev 1.4   15 Aug 1992 12:24:42   JED
 * Driver Version 4.01.38
   Added extern "C".
 *
 *     Rev 1.3   10 Aug 1992 13:09:10   JED
 * Driver Version 4.01.36
   Removed nPassThru from SDARG_State_Recogx; it is not needed and
   it made the block get out of sync with the non-x version.
 *
 *     Rev 1.2   10 Aug 1992 10:14:10   JED
 * Driver Version 4.01.35
   Added recogx, which is an inhouse version of the recog routines.
   The first enhancement is that it supports the pass thru list (words
   to be passed thru prefiltering).
   It also returns bestScore, the score of the best word.
 *
 *     Rev 1.1   27 Jul 1992 11:04:16   DEAN
 * No new version needed.
   Made all pointers in the declaration far.
 *
 *     Rev 1.0   27 Jul 1992 09:56:00   DEAN
 * Driver version 4.01.20
*/
#ifndef _SDAPIINH_H_
#define _SDAPIINH_H_

#ifdef __cplusplus
extern "C" {
#endif
```

```
#ifndef _SDAPI_H
#include "sdapi.h"
#endif

SD_UIT SD_ENTRYPT SDUtt_MakeRandom( SD_WORD hWord,
                                    SD_VOC hVoc );

uint SD_ENTRYPT  SDPar_GetNumPar( void );

SD_USER SD_ENTRYPT SDUser_Open( const char far *pFileName );

SD_VOC  SD_ENTRYPT SDVoc_Open( const char far *pVocFileName,
                               const char far *pMode );

//----------------------------------------------------------------------------
//                          From CHANNEL.C
//
//----------------------------------------------------------------------------

STRUC( SD_CHANNEL_INFO ) {
    bool isOpen;            /* YES if the channel is open            */
    bool isClaimed;         /* YES if this channel is claimed        */
    bool isMicOn;           /* YES if the mic is on for this channel */
    bool isMicSwitchOn;     /* YES if the phys mic sw is on          */
    bool isSpeechAvailable; /* YES if speech is available            */
    SD_CHANNEL_TYPE type;
};

void    SD_ENTRYPT SDChannel_GetInfo( SD_CHANNEL hChan,
                                      SD_CHANNEL_INFO far *pInfo );

//----------------------------------------------------------------------------
//                          From STATE.C
//
//----------------------------------------------------------------------------

STRUC( RECOG_PARAMETERS ) {
    int16  maxChoices;
    int16  rejThresh;
    int16  distThresh;
    int16  useLm;
    int16  computation;
    SD_CONTEXT far *pContext;
    char       far *prefix;
};

typedef uns32 SD_STATE_MACHINE;

/* The types of transitions defined are:

    UNDEFINED   No transition has been defined.  This can be used for
                words and states for which the default transition should
                be used.  The default transition is determined by searching
                recursively up the including states until a defined transition
                is found.  If no defined transition is found, it is treated
                as a RET.

    NOP         Stay in the current recognition state.

    JMP         Go to a specific state.

    CALL        Call a state, pushing the specified return state onto the
                stack.  When a return is executed, the return state becomes
                the current recognition state.
```

SDAPIINN.H 7-22-95 2:08p

```
    RET         Pop the next state off of the stack.  States are pushed onto
                the stack via CALL and CALL_NOP transitions.

    CALL_NOP    Call a state, pushing the current recognition state onto the
                stack.  When a return is executed, the current state becomes
                again the recognition state.

    CALL_GOTO   This opcode is only valid for included states. This causes
                the call to occur BEFORE considering transitions specific
                to the word.
*/

#define UNDEFINED       0
#define NOP             1
#define JMP             2
#define CALL            3
#define RET             4
#define CALL_NOP        5
#define CALL_GOTO       6
typedef uns16 SDFsgOpCode;

STRUC( SD_TRANSITION ) {
    SDFsgOpCode OpCode;
    union {

        struct {
            SD_STATE callState;
            SD_STATE returnState;
        } call;

        struct {
            SD_STATE destState;
        } jmp;

        struct {
            SD_STATE callState;
        } call_nop;

        struct {
            SD_STATE gotoState;
        } call_goto;

    } args;
};

STRUC( SD_STATE_INFO ) {
    int32          nWords;       /* current number of words in the state    */
    int32          nActive;      /* number of active words                  */
    SD_TRANSITION  Transition;   /* transition associated with state itself */
    SD_STATE       hParent;      /* parent state or 0                       */
    int16          nChildStates; /* number of child states                  */
    int16          nSubStates;   /* number of included substates            */
    int16          nEnv;         /* number of environment variables         */
    int16          wordOrder;    /* flags indicating the word orderings avail */
};

STRUC( SD_STATE_WORD_INFO ) {
    BOOL16 isInState;
    BOOL16 isActive;
    SD_TRANSITION Transition;
    int16 nEnv;
};

void SD_ENTRYPT SDState_GetInfo( SD_VOC hVoc,
                                 SD_STATE hState,
                                 SD_STATE_INFO far *pInfo );
```

Page 2 of 5

```
void SD_ENTRYPT SDState_GetWordInfo( SD_VOC hVoc,
                                     SD_STATE hState,
                                     SD_WORD hWord,
                                     SD_STATE_WORD_INFO far *pInfo );

SD_TRANSITION SD_ENTRYPT SDState_GetTransition( SD_VOC hVoc,
                                                SD_STATE hState );

void SD_ENTRYPT SDState_SetTransition( SD_VOC hVoc,
                                       SD_STATE hState,
                                       SD_TRANSITION trans );

SD_TRANSITION SD_ENTRYPT SDState_GetWordTransition( SD_VOC hVoc,
                                                    SD_STATE hState,
                                                    SD_WORD hWord );

void SD_ENTRYPT SDState_SetWordTransition( SD_VOC hVoc,
                                           SD_STATE hState,
                                           SD_WORD hWord,
                                           SD_TRANSITION trans );

SD_TRANSITION SD_ENTRYPT SDState_GetStateTransition( SD_VOC hVoc,
                                                     SD_STATE hState,
                                                     SD_STATE hSubState );

void SD_ENTRYPT SDState_SetStateTransition( SD_VOC hVoc,
                                            SD_STATE hState,
                                            SD_STATE hSubState,
                                            SD_TRANSITION trans );

int SD_ENTRYPT SDState_Recog( SD_VOC hVoc,
                              SD_STATE hState,
                              SD_UTT hUtt,
                              RECOG_RESULT_ENTRY far *pResults,
                              size_t lResults,
                              RECOG_PARAMETERS far *pPar,
                              RECOG_STATUS far *pRecStatus );

int SD_ENTRYPT SDState_ContRecog( SD_VOC hVoc,
                                  SD_STATE_MACHINE hStMach,
                                  SD_UTT hUtt,
                                  RECOG_RESULT_ENTRY far *pResults,
                                  size_t lResults,
                                  RECOG_PARAMETERS far *pPar,
                                  RECOG_STATUS far *pRecStatus );

void SD_ENTRYPT SDState_ContRecogPartial( RECOG_RESULT_ENTRY *pResults,
                                          size_t lResults );


//----------------------------------------------------------------------
//                        STATE MACHINE MANAGEMENT
//
//----------------------------------------------------------------------

SD_STATE_MACHINE SD_ENTRYPT SDStMach_New(SD_VOC hVoc, SD_STATE hState);

void SD_ENTRYPT SDStMach_Delete( SD_STATE_MACHINE hStMach );

SD_STATE_MACHINE SD_ENTRYPT SDStMach_Update( SD_VOC hVoc,
                                             SD_STATE_MACHINE hMachIn,
                                             SD_WORD hWord,
                                             bool delMachIn );

#if 0
```

SDAPIINN.H  7-22-95  2:08p

```
SD_STATE SD_ENTRYPT SDStMach_GetCurrentState( SD_STATE_MACHINE hStMach );
#endif

// 1st implementation - ParseBuf will contain nWords 0-terminated arrays
// of SD_STATE_SPECs
bool SD_ENTRYPT SDStMach_Parse(SD_STATE_MACHINE hMachIn,
                               SD_WORD_SPEC far *pWordSpecs,
                               UINT nWords,
                               void *pParseBuf, size_t lParseBuf,
                               size_t *plParse);

//----------------------------------------------------------------------
//                        From LEARN.C
//
//----------------------------------------------------------------------

void SD_ENTRYPT SDWord_Learn( SD_VOC                     hVoc,
                              int                        nWords,
                              const char          far *pName,
                              SD_UTT                     hUtt,
                              RECOG_RESULT_ENTRY far *pResults,
                              RECOG_PARAMETERS    far *pRecPar,
                              RECOG_STATUS        far *pRecStatus );


//----------------------------------------------------------------------
//                        From TRAIN.C
//
//----------------------------------------------------------------------

STRUC( TRAIN_PARAMETERS ) {
    int32  weight;
    int16  tolerance;
};

void SD_ENTRYPT SDWord_Train( SD_VOC                     hVoc,
                              int                        nWords,
                              const char        far *pName,
                              int                        nUtt,
                              SD_UTT            far *pUtt,
                              TRAIN_PARAMETERS far *pPar,
                              TRAIN_STATUS      far *pStatus,
                              SD_REJCODE        far *pRejCodes );


//----------------------------------------------------------------------
//                        From ADPTWRD.C
//
//----------------------------------------------------------------------

void SD_ENTRYPT SDWord_Adapt( SD_VOC                     hVoc,
                              int                        nWords,
                              const char        far *pName,
                              int                        nUtt,
                              SD_UTT            far *pUtt,
                              TRAIN_PARAMETERS far *pPar,
                              TRAIN_STATUS      far *pStatus,
                              SD_REJCODE        far *pRejCodes );


//----------------------------------------------------------------------
//                        From ADPTLMIO.C
//
//----------------------------------------------------------------------

STRUC( ADAPTLM_PARAMETERS ) {
```

Page 3 of 5

```
    int32           longTermRelevance;
    int32           shortTermRelevance;
    SD_CONTEXT far *pContext;
);

void SD_ENTRYPT SDWord_AdaptLm( SD_VOC hVoc,
                                SD_WORD hWord,
                                ADAPTLM_PARAMETERS far *pPar );


//-------------------------------------------------------------------
//                    From CALLD.C
//
//-------------------------------------------------------------------
int SD_ENTRYPT SDCallDriver( int funcId,
                             void far *pArgs,
                             size_t lArgs,
                             int nPtrs );


//-------------------------------------------------------------------
//                    From GFUNCID.C
//
//-------------------------------------------------------------------

int SD_ENTRYPT SDGetFuncId( const char far *pName );

int SD_ENTRYPT SDGetFuncName( int funcId,
                              char far *pName,
                              size_t lBuf );

typedef void ( SD_CALLBACK *SDAPI_FUNC_CALLBACK )
                  ( int funcid, SD_TASK hTask, void far *pargs );


int SD_ENTRYPT SDSetFuncCallBack( SDAPI_FUNC_CALLBACK EntryFunc,
                                  SDAPI_FUNC_CALLBACK ExitFunc );


/*******************************************************************
**
** These functions were never available in SDAPI.H in the past.
*/

/* from lm.c: */
void SD_ENTRYPT SDLm_Score(SD_VOC hVoc, int32 nWords, SD_WORD far *pWords,
  SD_CONTEXT far *pContext, uns16 far *pScores);


bool SD_ENTRYPT SDLm_Reorder(int nStates,
                             SD_STATE_SPEC far *pStates,
                             int nChoicesL,
                             RECOG_RESULT_ENTRY far *pResultsL,
                             SD_CONTEXT far *pContextL,
                             int nChoicesR,
                             RECOG_RESULT_ENTRY far *pResultsR,
                             SD_CONTEXT far *pContextR);

// has to agree with RECOG_STATUS
STRUC( RECOGX_STATUS ) {
    SD_REJCODE rejCode;
    int16 confidence;
    int16 nChoices;
    uns16 bestScore;    // score of top choice
);
```

SDAPIINN.H 7-22-95 2:08p

```
// has to agree with RECOG_PARAMETERS
STRUC( RECOGX_PARAMETERS ) {
    int16 maxChoices;
    int16 rejThresh;
    int16 distThresh;
    int16 useLm;
    int16 computation;
    SD_CONTEXT far *pContext;
    char far *prefix;
    SD_WORD far *pPassThru;
    uns16 nPassThru;
);

int SD_ENTRYPT SDState_Recogx( SD_VOC hVoc,
                               SD_STATE state,
                               SD_UTT hUtt,
                               RECOG_RESULT_ENTRY far *pResults,
                               size_t lResults,
                               RECOGX_PARAMETERS far *pPar,
                               RECOGX_STATUS far *pRecStatus );

uint SD_ENTRYPT SDWord_ListIds( SD_VOC hVoc,
                                SD_WORD hWord,
                                SD_WORD far *pBuf,
                                uns32 nBuf );


size_t SD_ENTRYPT SDWord_GetPronunciation( SD_VOC hVoc,
                                           SD_WORD hWord,
                                           unsigned char far *pBuf,
                                           size_t lBuf );

uns32 SD_ENTRYPT SDWord_GetLmlCount( SD_VOC hVoc,
                                     SD_WORD hWord );

void SD_ENTRYPT SDWord_SetLmlCount( SD_VOC hVoc,
                                    SD_WORD hWord,
                                    uns32 lmlCount );

uns32 SD_ENTRYPT SDWord_GetLmlCountWithRecent( SD_VOC hVoc,
                                               SD_WORD hWord );

size_t SD_ENTRYPT SDUtt_Get( SD_UTT hUtt,
                             void far *pBuf,
                             size_t lBuf );
size_t SD_ENTRYPT SDUtt_GetI( SD_UTT hUtt,
                              void far *pBuf,
                              size_t lBuf );

STRUC( SD_VOC_EXTENDED_INFO ) {
    uns32 nIds;       /* current number of Ids in the vocabulary */
    uns32 nIdsResident;     /* number of resident Ids */
);
void SD_ENTRYPT SDVoc_GetExtendedInfo( SD_VOC hVoc,
                                       SD_VOC_EXTENDED_INFO far *pInfo );

STRUC( SD_STATE_EXTENDED_INFO ) {
    uns32 nIds;       /* current number of Ids in the state */
);

void SD_ENTRYPT SDState_GetExtendedInfo( SD_VOC hVoc,
                                         SD_STATE hState,
                                         SD_STATE_EXTENDED_INFO far *pInfo );

int SD_ENTRYPT SDState_PhonRecog( SD_VOC hVoc,
                                  SD_UTT hUtt,
```

Page 4 of 5

```
                                    RECOG_RESULT_ENTRY far *pResults,
                                    size_t lResults,
                                    RECOG_PARAMETERS far *pPar,
                                    RECOG_STATUS far *pRecStatus,
                                    SD_WORD *pTranscription,
                                    int16 nTranscription );


// Calls the score decay for the lMadjustments that ordinarily happens only
// during the load/save cycle.  (GRG, 1SEP93)
void SD_ENTRYPT SDLm_DecayAdjust( SD_VOC hVoc );


// Categorizes the list of words in a given position  (GRG, 1SEP93)
// (pBufs points to a contiguous 2-D array of chars)
void SDLm_GetCats( SD_VOC hVoc,
                   int32 nWords,
                   SD_WORD *pWords,
                   int16 posPrior,
                   char far *pBufs[],
                   size_t lBufs );


#ifdef __cplusplus
}   /* extern "C" */
#endif

#endif /* _SDAPIINN_H_ */
```

SDAPIINN.H  7-22-95  2:08p

```
/* $Header:   F:/work/newd/inc/vcs/sdapix.h_v   1.45   21 Feb 1994 09:37:44   JE
  => D  $ */
/* FILE: sdapix.h

    Contains internal definitions for the sdapi library.

    Copyright (c) Dragon Systems,Inc. 1992-1994

    AUTHOR: Jed Roberts
    CREATED: 18-Mar-1992

    MODIFICATIONS
    $Log:   F:/work/newd/inc/vcs/sdapix.h_v  $

     Revision 6 on Fri Apr 29 20:13:23 1994 by JED
    Driver version 4.13.50
    Added SDARG_User_NotifyFileRename & SDARG_Voc_NotifyFileRename

     Revision 5 on Tue Apr 19 17:52:03 1994 by JED
    Driver version 4.13.40
    Fixed SDARG_Lm_DecayAdjust and SDARG_Lm_GetCats to have header field.

     Revision 4 on Thu Apr 14 16:34:42 1994 by DEAN
    Driver Version 4.13.38
    Changed the name of some arguments to StMach_Update.
    Added SDARG_StMach_Parse.

     Revision 3 on Thu Apr 07 18:44:37 1994 by JED
    Driver version 4.13.35
    Added MAP_SD_BUFFER macro.
    Added SDARG_Voc_GetWeight & SDARG_Voc_SetWeight.
    Added SDARG_Word_CopyToUser.
    Added SDARG_Lm_Reorder.
    Added SDARG_State_Recog2.
    int -> int16
    size_t -> uns16

     Revision 2 on Wed Mar 30 18:55:55 1994 by DEAN
    Driver Version 4.13.33 - cleanup of sdapi*.h
    Now this file contains ALL SDARG structures, inhouse or not.
    Added conditional includes of sdapi.h and sdapiinh.h.

     Revision 1 on Thu Mar 17 13:20:16 1994 by DEAN
    Driver Version 4.13.25 - first version under TLIB!
 *
 *    Rev 1.45   21 Feb 1994 09:37:44   JED
 * Driver version 4.13.15
    Added SDARG_Word_HasModel and SDARG_Word_StateRefCount.
 *
 *    Rev 1.44   04 Feb 1994 17:13:02   BETSY
 * Driver Version 4.13.13
    Fixed order of SDARG_Word_BuildFrom
    Buddy=Jed.
 *
 *    Rev 1.43   31 Jan 1994 18:39:20   JED
 * Driver version 4.13.11
    Buddy==Frank.
    Added SDARG_State_CopyFrom.
 *
 *    Rev 1.42   27 Jan 1994 19:24:36   BETSY
 * Driver Version 4.13.10
 * Macintosh Changes - Really IAN - Buddy = DEAN = GREG
 *
    Macintosh - removed unnecessary mac specific definitions.
 *
 *    Rev 1.41   16 Jan 1994 17:32:38   JED
```

```
 * Driver version 4.13.4
    Added SDARG_Word_ListMatchesByFrequency.
 *
 *    Rev 1.40   13 Jan 1994 12:08:00   BETSY
 * Driver Version 4.13.3
    Added SDARG_State_IterateSpecifiedWords and SDARG_State_NextSpecifiedWords.
 *
 *    Rev 1.39   01 Sep 1993 20:06:16   DEAN
 * Driver Version 4.11.131
    Added SDARG_Lm_DecayAdjust and SDARG_Lm_GetCats (GRG)
 *
 *    Rev 1.38   31 Aug 1993 11:19:08   DEAN
 * Driver Version 4.11.127 (GRG's language model stuff)
    Added SDARG_Voc_SetLmType1   (GRG)
 *
 *    Rev 1.37   23 Aug 1993 17:42:22   FRANKM
 * Driver Version 4.10.34
    Added SDARG_Word_BuildFrom
 *
 *    Rev 1.36   16 Aug 1993 21:17:36   BETSY
 * Driver Version 4.11.110
    Added definition of EXTERN.
 *
 *    Rev 1.35   09 Aug 1993 13:27:50   BETSY
 * Driver Version 4.11.103
    Added SETUP_SD_INPUT_FILESPEC.
 *
 *    Rev 1.34   28 Jul 1993 12:03:16   BETSY
 * Driver Version 4.11.93
    Added args fo SDUtt_GetLabel, SDUtt_SetLabel and
    SDUtt_Compare.
 *
 *    Rev 1.33   23 Jun 1993 12:02:28   BETSY
 * Driver Version 4.11.64
 *
 *    Rev 1.32   09 Jun 1993 14:53:14   BETSY
 * Driver Version 4.11.52
 *
 *    Rev 1.31   20 May 1993 07:10:16   JOELG
 * Driver version 4.11.39
    Added maxWords and minWords to the SDARG_State_ContRecog1 block.
 *
 *    Rev 1.30   19 May 1993 16:00:06   JED
 * Driver Version 4.11.35
    Added SDARG_Voc_GetCollationTables and SDARG_Voc_SetCollationTables.
 *
 *    Rev 1.29   19 May 1993 09:14:58   DEAN
 * Driver Version 4.11.32
    Redefined the STRUC macro.
 *
 *    Rev 1.28   17 May 1993 16:21:18   JOELG
 * Driver version 4.11.29
 *   added SDARG_Channel_GetInfo1 and changed the name of
 *   SDARG_Utt_Cancel, SDARG_Utt_SetWatchDog and SDArg_Api_GetVersion
 *
 *    Rev 1.27   17 May 1993 11:50:02   JOELG
 * Driver version 4.11.26
    Added support for small model compiles of the SDAPI interface
    by using _fstrlen instead of strlen in a macro is necessary.
 *
 *    Rev 1.26   13 May 1993 19:28:48   JED
 * Driver Version 4.11.22
    Added useRealModeCallbacks argument to SDARG_Task_New.
 *
 *    Rev 1.25   06 May 1993 12:15:18   BORIS
 * Driver Version 4.11.08
```

```
   Added new function blocks for the functions added in the toolkit,
   including SDPar_Iterate, SDState_ContRecog).
 *
 *    Rev 1.24   08 Apr 1993 21:20:38   DEAN
 * Driver Version 4.10.61 - new SD_BUFFER_SPEC
   New buffer spec (SD_BUFFER_SPEC2) - use with SDARG_TYPE = 2.
   New function id SDID_GET_MAX_SDARG_TYPE.
 *
 *    Rev 1.23   11 Mar 1993 14:54:52   JOSHUA
 * Driver version 4.10.42
 * Phoneme recognizer, minor bug fixes, ContRecogPartial.
   Added SDARG_ContRecogPartial.
 *
 *    Rev 1.22   27 Feb 1993 13:19:28   DEAN
 * Driver Version 4.10.39
   32-bit bug fix: int -> int16 in argument blocks
 *
 *    Rev 1.21   24 Feb 1993 15:18:20   JOSHUA
 * Driver version 4.10.36
 * Phoneme recognizer and minor bug fixes.
 *
 *

   Added transcription parameters to ContRecog
 *
 *    Rev 1.20   20 Feb 1993 17:18:32   JED
 * Driver Version 4.10.34
   Added SDUser_ForceResident and SDUser_DeleteWord.
 *
 *    Rev 1.19   17 Jan 1993 11:33:42   DEAN
 * Driver Version 4.10.24 - 32-bit cleanup part 3
 *
 *
 *    Rev 1.18   02 Nov 1992 10:19:14   JED
 * Driver Version 4.04.24
   Added SDVoc_GetExtendedInfo & SDState_GetExtendedInfo.
 *
 *    Rev 1.17   21 Sep 1992 16:35:06   JED
 * Driver Version 4.03.19
   Added Channel_SetLength.
 *
 *    Rev 1.16   16 Sep 1992 15:15:00   JED
 * Driver Version 4.03.15
   Added args for SDQuit and SDSetWatchDog.
 *
 *    Rev 1.15   13 Sep 1992 14:40:20   JED
 * Driver Version 4.03.11
   Added SDARG_Voc_SetLmType.
 *
 *    Rev 1.14   11 Sep 1992 12:41:18   JED
 * Driver Version 4.03.10
   Added SDARG_Error_GetMessage2 which adds hTask, funcid and errorCode
   to SDARG_Error_GetMessage so that we can
   distinguish between root and task functions and return the correct
   error message.
 *
 *    Rev 1.13   04 Sep 1992 13:36:24   JOSHUA
 * Driver version 4.02.12
 *
 *    Rev 1.12   23 Aug 1992 16:49:30   JED
 *
 *    Rev 1.11   15 Aug 1992 12:25:04   JED
 * Driver Version 4.01.38
   No longer require std.h.
 *
 *    Rev 1.10   05 Aug 1992 19:14:10   JOSHUA
 * Driver version 4.01.30
```

```
   Added STRUC declaration and re-declared SDARG_State_Recog with
   it.  (This is needed so that csr.h doesn't have to include sdapix.h)
 *
 *    Rev 1.9   26 Jul 1992 15:09:44   JED
 * Driver Version 4.01.18
 *
 *    Rev 1.8   20 Jul 1992 12:07:36   JED
 * Driver Version 4.01.14
   Added SDARG_Word_Lookup, SDARG_Word_List.
 *
 *    Rev 1.7   16 Jul 1992 16:56:58   JOEL
 * Minor fixes to SDAPI Entry and Exit callback defs
 *
 *    Rev 1.6   16 Jul 1992 16:35:10   JOEL
 * Modified to support addtion of SDGetFuncName and Func Callbacks in func.cpp
 *
 *    Rev 1.5   11 Jul 1992 14:55:26   JED
 * Driver Version 4.01.9
 *
 *    Rev 1.4   10 Jul 1992 15:53:38   JED
 * Driver Version 4.01.7
   Added SDARG_State_GetInfo.
   Addes SDARG_State_GetWordInfo.
 *
 *    Rev 1.3   06 Jul 1992 18:23:24   JED
 * Driver Version 4.01.6
 *
 *    Rev 1.2   08 May 1992 17:50:26   JED
 * Version 4.00.9
   Cosmetic name changes of SDARG_ structs.
   Added SDChannel_GetStatus.
 *
 *    Rev 1.1   09 Apr 1992 09:27:06   JED
 *
 *    Rev 1.0   25 Mar 1992 20:03:50   JED
 *    ......................................


*/

#ifndef _SDAPIX_H_
#define _SDAPIX_H_

#ifndef _SDAPI_H
#include "sdapi.h"
#endif

#ifndef _SDAPIINH_H
#include "sdapiinh.h"
#endif

#ifndef PUBLIC
#define PUBLIC
#endif

#ifndef EXTERN
#ifdef __cplusplus
#define EXTERN extern "C"
#else
#define EXTERN extern
#endif
#endif

#ifndef STRUC
#define STRUC(x) typedef struct x x; struct x
    /* structures defined using this macro can be referred to
```

```
        without the 'struct' qualifier */
#endif

//#include "filespec.h"

/* define the current argument structure type */
extern unsigned int SDARG_TYPE;      /* app-side global variable in INIT.C */
#define MAX_SDARG_TYPE 2

/* Structure used to define a buffer of info to be passed to the speech
   driver */
// OLD SPEC:
typedef struct SD_BUFFER_SPEC1 {
    union {
        void far *p;
    uns32 absadrs;
    } pBuf;
    uns32 lBuf;
} SD_BUFFER_SPEC1;
// NEW SPEC:
typedef struct SD_BUFFER_SPEC2 {
    union {
        void far *p;
    uns32 absadrs;
    } pBuf;
    uns32 lBuf;
    uns32 nDataBytes;        /* # of bytes of data in buffer */
} SD_BUFFER_SPEC2;

typedef SD_BUFFER_SPEC2 SD_BUFFER_SPEC;

#define SETUP_SD_BUFFER(_b,_p,_l,_n) \
    (_b).pBuf.p = (void far *)(_p); \
    (_b).lBuf = _l; \
    (_b).nDataBytes = _n;

/* if buffer is filled with data, you can use SETUP_SD_IBUFFERX */
#define SETUP_SD_INPUT_BUFFER(_b,_p,_l) \
    (_b).pBuf.p = (void far *)(_p); \
    (_b).nDataBytes = (_b).lBuf = _l;

#define SETUP_SD_OUTPUT_BUFFER(_b,_p,_l) SETUP_SD_BUFFER(_b,_p,_l,0)
#define SETUP_SD_IO_BUFFER(_b,_p,_l) SETUP_SD_INPUT_BUFFER(_b,_p,_l)

#define SETUP_SD_INPUT_FILESPEC(_b,_p) \
    SETUP_SD_INPUT_BUFFER(_b,_p,(_p) ? FileSpec_GetSize(_p)+1 : 0)

#if defined(__SMALL__)
#define SETUP_SD_INPUT_STRING(_b,_p) \
    SETUP_SD_INPUT_BUFFER(_b,_p,(_p) ? _fstrlen(_p)+1 : 0)
#else
#define SETUP_SD_INPUT_STRING(_b,_p) \
    SETUP_SD_INPUT_BUFFER(_b,_p,(_p) ? strlen(_p)+1 : 0)
#endif

/* macro to pull the pointer out of a buffer spec */
#define MAP_SD_BUFFER(T,b) ((T *)((b).pBuf.p))

/* _p must be a SD_CONTEXT * for the following two macros */
#define SD_CONTEXT_LEN(_p) \
((_p) ? sizeof((_p)->nContext) + (_p)->nContext * sizeof((_p)->context[0]) : 0)
#define SETUP_SD_CONTEXT(_b,_p) SETUP_SD_INPUT_BUFFER(_b,_p,SD_CONTEXT_LEN(_p))

/* Predefined function IDs: */
#define SDID_GET_FUNC_ID 1
#define SDID_GET_MAX_SDARG_TYPE (0xfffd)
```

SDAPIX.H 7-22-95 2:09p

```
#define SD_MAX_NPTRARGS 6            // max # of ptr args to any driver call

typedef struct {
    uns16 type;
    uns16 OffsetToPtrs; // offset from start of arg block (incl. arg header)
    uns16 nPtrArgs;
} SDARG_HEADER;

typedef struct {
    SDARG_HEADER h;
    int16 ret;
    SD_BUFFER_SPEC pName;
} SDARG_GetFuncId;

typedef struct {
    SDARG_HEADER h;
    int16 FuncId;
    uns16 ret;
    SD_BUFFER_SPEC pName;
} SDARG_GetFuncName;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_BUFFER_SPEC pBuf;
} SDARG_Api_GetVersion;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_BUFFER_SPEC pBuf;
} SDARG_Error_GetMessage;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_TASK hTask;
    int16  funcid;
    int16  errorCode;
    SD_BUFFER_SPEC pBuf;
} SDARG_Error_GetMessage2;

typedef struct {
    SDARG_HEADER h;
    void (SD_CALLBACK *ret)(void);
#ifndef _MACINTOSH_
    uns16 ds;
#else
    long A5;
#endif
    void (SD_CALLBACK *pfunc)(void);
} SDARG_Utt_SetWatchDog;

typedef struct {
    SDARG_HEADER h;
} SDARG_Utt_Cancel;

typedef struct {
    SDARG_HEADER h;
    SD_TASK ret;
    SD_BUFFER_SPEC pName;
} SDARG_Task_New_Obsolete;

typedef struct {
    SDARG_HEADER h;
```

Page 3 of 15

```
    SD_TASK ret;
    uns16 useRealModeCallbacks;
    SD_BUFFER_SPEC pName;
} SDARG_Task_New;

typedef struct {
    SDARG_HEADER h;
    SD_TASK hTask;
} SDARG_Task_Close;

typedef struct {
    SDARG_HEADER h;
    SD_TASK hTask;
    uns16 ret;
    SD_BUFFER_SPEC pBuf;
} SDARG_Task_GetName;

typedef struct {
    SDARG_HEADER h;
    SD_TASK ret;
    SD_BUFFER_SPEC pName;
} SDARG_Task_GetHandle;

typedef struct {
    SDARG_HEADER h;
    SD_BUFFER_SPEC pIter;
} SDARG_Task_Iterate;

typedef struct {
    SDARG_HEADER h;
    SD_TASK ret;
    SD_BUFFER_SPEC pIter;
} SDARG_Task_Next;

typedef struct {
    SDARG_HEADER h;
    SD_CHANNEL ret;
    SD_BUFFER_SPEC pFileName;
    SD_BUFFER_SPEC pMode;
} SDARG_Channel_NewFile;

typedef struct {
    SDARG_HEADER h;
    SD_CHANNEL hChan;
} SDARG_Channel_Open;

typedef struct {
    SDARG_HEADER h;
    SD_CHANNEL hChan;
} SDARG_Channel_Close;

typedef struct {
    SDARG_HEADER h;
    SD_UIT ret;
    SD_CHANNEL hChan;
    int16  readMode;
} SDARG_Channel_Read;

typedef struct {
    SDARG_HEADER h;
    SD_CHANNEL hChan;
    SD_UIT hUit;
} SDARG_Channel_Write;

typedef struct {
    SDARG_HEADER h;
```

```
    int32 ret;
    SD_CHANNEL hChan;
    int32 nUit;
    int16  seekMode;
} SDARG_Channel_Seek;

typedef struct {
    SDARG_HEADER h;
    SD_CHANNEL hChan;
    int32 nUit;
} SDARG_Channel_SetLength;

typedef struct {
    SDARG_HEADER h;
    SD_BUFFER_SPEC pIter;
} SDARG_Channel_Iterate;

typedef struct {
    SDARG_HEADER h;
    SD_CHANNEL ret;
    SD_BUFFER_SPEC pIter;
} SDARG_Channel_Next;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_CHANNEL hChan;
    SD_BUFFER_SPEC pBuf;
} SDARG_Channel_GetName;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_CHANNEL hChan;
    SD_BUFFER_SPEC pBuf;
} SDARG_Channel_GetDeviceName;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_CHANNEL hChan;
    SD_BUFFER_SPEC pBuf;
} SDARG_Channel_GetDeviceRev;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_CHANNEL hChan;
    SD_BUFFER_SPEC pBuf;
} SDARG_Channel_GetDeviceAddress;

typedef struct {
    SDARG_HEADER h;
    int16  ret;
    SD_CHANNEL hChan;
    int16  seize;
} SDARG_Channel_Claim;

typedef struct {
    SDARG_HEADER h;
    SD_CHANNEL hChan;
} SDARG_Channel_Release;

typedef struct {
    SDARG_HEADER h;
    SD_CHANNEL hChan;
    SD_BUFFER_SPEC pInfo;
) SDARG_Channel_GetInfo;

typedef struct (
    SDARG_HEADER h;
    SD_CHANNEL hChan;
    SD_BUFFER_SPEC pInfo;
) SDARG_Channel_GetInfo1;

typedef struct (
    SDARG_HEADER h;
    int16 ret;
    SD_CHANNEL hChan;
) SDARG_Channel_IsClaimed;

typedef struct (
    SDARG_HEADER h;
    int16 ret;
    SD_CHANNEL hChan;
    int16  purge;
    int16  micSwitch;
) SDARG_Channel_SetMicOn;

typedef struct (
    SDARG_HEADER h;
    int16 ret;
    SD_CHANNEL hChan;
) SDARG_Channel_SetMicOff;

typedef struct (
    SDARG_HEADER h;
    int16 ret;
    SD_CHANNEL hChan;
) SDARG_Channel_IsMicOn;

typedef struct (
    SDARG_HEADER h;
    int16 ret;
    SD_CHANNEL hChan;
) SDARG_Channel_IsMicSwitchOn;

typedef struct (
    SDARG_HEADER h;
    int16 ret;
    SD_CHANNEL hChan;
) SDARG_Channel_IsSpeechAvailable;

typedef struct (
    SDARG_HEADER h;
    void (SD_CALLBACK *ret)(SD_CHANNEL, SD_CHANNEL_EVENT);
    SD_CHANNEL hChan;
    int16  event;
#ifndef _MACINTOSH_
    uns16 ds;
#else
    long A5;
#endif
    void (SD_CALLBACK *pHandler)(SD_CHANNEL, SD_CHANNEL_EVENT);
) SDARG_Channel_SetEventHandler;

typedef struct (
    SDARG_HEADER h;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pInfo;
) SDARG_Utt_Collect;
```

SDAPIX.H 7-22-95 2:09p

```
typedef struct (
    SDARG_HEADER h;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pInfo;
) SDARG_Utt_GetInfo;

typedef struct (
    SDARG_HEADER h;
    SD_UTT hUtt;
) SDARG_Utt_Delete;

typedef struct (
    SDARG_HEADER h;
    SD_BUFFER_SPEC pIter;
) SDARG_Utt_Iterate;

typedef struct (
    SDARG_HEADER h;
    SD_UTT ret;
    SD_BUFFER_SPEC pIter;
) SDARG_Utt_Next;

typedef struct (
    SDARG_HEADER h;
    uns32 ret;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pBuf;
) SDARG_Utt_GetLabel;

typedef struct (
    SDARG_HEADER h;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pUttLabel;
) SDARG_Utt_SetLabel;

typedef struct (
    SDARG_HEADER h;
    uns32 ret;
    SD_UTT hUtt1;
    SD_UTT hUtt2;
) SDARG_Utt_Compare;

typedef struct (
    SDARG_HEADER h;
    SD_USER ret;
    SD_BUFFER_SPEC pFileName;
) SDARG_User_Open;

typedef struct (
    SDARG_HEADER h;
    SD_USER ret;
) SDARG_User_New;

typedef struct (
    SDARG_HEADER h;
    SD_USER hUser;
) SDARG_User_Save;

typedef struct (
    SDARG_HEADER h;
    SD_USER hUser;
    SD_BUFFER_SPEC pFileName;
) SDARG_User_SaveAs;

typedef struct (
    SDARG_HEADER h;
```

Page 5 of 15

```
        SD_USER hUser;
    } SDARG_User_Close;

    typedef struct {
        SDARG_HEADER h;
        SD_USER hUser;
        SD_BUFFER_SPEC pInfo;
    } SDARG_User_GetInfo;

    typedef struct {
        SDARG_HEADER h;
        uns16 ret;
        SD_USER hUser;
        SD_BUFFER_SPEC pBuf;
    } SDARG_User_GetFileName;

    typedef struct {
        SDARG_HEADER h;
        SD_USER hUser;
        SD_BUFFER_SPEC pFileName;
    } SDARG_User_SetFileName;

    typedef struct {
        SDARG_HEADER h;
        SD_USER hUser;
        SD_BUFFER_SPEC pNewFileSpec;
    } SDARG_User_NotifyFileRename;

    typedef struct {
        SDARG_HEADER h;
        SD_BUFFER_SPEC pIter;
    } SDARG_User_Iterate;

    typedef struct {
        SDARG_HEADER h;
        SD_USER ret;
        SD_BUFFER_SPEC pIter;
    } SDARG_User_Next;

    typedef struct {
        SDARG_HEADER h;
        SD_USER ret;
        SD_USER hUser;
    } SDARG_User_SetCurrent;

    typedef struct {
        SDARG_HEADER h;
        SD_USER ret;
    } SDARG_User_GetCurrent;

    typedef struct {
        SDARG_HEADER h;
        uns32 ret;
        SD_USER hUser;
        SD_BUFFER_SPEC pName;
        SD_BUFFER_SPEC pBuf;
    } SDARG_User_GetEnv;

    typedef struct {
        SDARG_HEADER h;
        SD_USER hUser;
        SD_BUFFER_SPEC pName;
        SD_BUFFER_SPEC pBuf;
    } SDARG_User_SetEnv;

    typedef struct {
```

SDAPIX.H 7-22-95 2:09p

```
        SDARG_HEADER h;
        SD_USER hUser;
        SD_BUFFER_SPEC pIter;
    } SDARG_User_IterateEnv;

    typedef struct {
        SDARG_HEADER h;
        uns32 ret;
        SD_BUFFER_SPEC pBuf;
        SD_BUFFER_SPEC pIter;
    } SDARG_User_NextEnv;

    typedef struct {
        SDARG_HEADER h;
        SD_USER hUser;
    } SDARG_User_ForceResident;

    typedef struct {
        SDARG_HEADER h;
        int16 ret;
        SD_USER hUser;
        SD_BUFFER_SPEC pName;
    } SDARG_User_DeleteWord;

    typedef struct {
        SDARG_HEADER h;
        SD_VOC ret;
    } SDARG_Voc_New;

    typedef struct {
        SDARG_HEADER h;
        SD_VOC ret;
        SD_BUFFER_SPEC pFileName;
        SD_BUFFER_SPEC pMode;
    } SDARG_Voc_Open;

    typedef struct {
        SDARG_HEADER h;
        SD_VOC hVoc;
    } SDARG_Voc_Save;

    typedef struct {
        SDARG_HEADER h;
        SD_VOC hVoc;
        SD_BUFFER_SPEC pFileName;
    } SDARG_Voc_SaveAs;

    typedef struct {
        SDARG_HEADER h;
        SD_VOC hVoc;
    } SDARG_Voc_Close;

    typedef struct {
        SDARG_HEADER h;
        SD_VOC hVoc;
        SD_BUFFER_SPEC pInfo;
    } SDARG_Voc_GetInfo;

    typedef struct {
        SDARG_HEADER h;
        SD_VOC hVoc;
        SD_BUFFER_SPEC pInfo;
    } SDARG_Voc_GetExtendedInfo;

    typedef struct {
        SDARG_HEADER h;
```

Page 6 of 15

```
    uns16 ret;
    SD_VOC hVoc;
    SD_BUFFER_SPEC pBuf;
} SDARG_Voc_GetFileName;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_BUFFER_SPEC pfileName;
} SDARG_Voc_SetFileName;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_BUFFER_SPEC pNewFileSpec;
} SDARG_Voc_NotifyFileRename;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_LM_TYPE lmType;
} SDARG_Voc_SetLmType;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hTargetVoc;
    SD_VOC hSourceVoc;
    SD_LM_TYPE lmType;
} SDARG_Voc_SetLmType1;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
} SDARG_Voc_GetWeight;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    uns32 weight;
} SDARG_Voc_SetWeight;

typedef struct {
    SDARG_HEADER h;
    SD_BUFFER_SPEC pIter;
} SDARG_Voc_Iterate;

typedef struct {
    SDARG_HEADER h;
    SD_VOC ret;
    SD_BUFFER_SPEC pIter;
} SDARG_Voc_Next;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pBuf;
} SDARG_Voc_GetEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pBuf;
```

```
} SDARG_Voc_SetEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_BUFFER_SPEC pIter;
} SDARG_Voc_IterateEnv;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_BUFFER_SPEC pBuf;
    SD_BUFFER_SPEC pIter;
} SDARG_Voc_NextEnv;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pResults;
    SD_BUFFER_SPEC pPar;
    SD_BUFFER_SPEC pRecStatus;
    SD_BUFFER_SPEC pContext;
    SD_BUFFER_SPEC prefix;
} SDARG_Voc_Recog;

typedef struct {
    SDARG_HEADER h;
    SD_BUFFER_SPEC pPass1Table;
    SD_BUFFER_SPEC pPass2Table;
} SDARG_Voc_GetCollationTables;

typedef struct {
    SDARG_HEADER h;
    SD_BUFFER_SPEC pPass1Table;
    SD_BUFFER_SPEC pPass2Table;
} SDARG_Voc_SetCollationTables;

typedef struct {
    SDARG_HEADER h;
} SDARG_Ref_Load;

typedef struct {
    SDARG_HEADER h;
} SDARG_Ref_Save;

typedef struct {
    SDARG_HEADER h;
    SD_WORD ret;
    SD_VOC hVoc;
    SD_BUFFER_SPEC pName;
} SDARG_Word_New;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
} SDARG_Word_Save;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
} SDARG_Word_Load;
```

```
typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
} SDARG_Word_Unload;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
} SDARG_Word_Delete;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_VOC hVoc;
    SD_BUFFER_SPEC pName;
} SDARG_Word_GetHandle;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_VOC hVoc;
    SD_WORD hWord;
    SD_BUFFER_SPEC pBuf;
} SDARG_Word_GetName;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
    SD_BUFFER_SPEC pNewName;
} SDARG_Word_SetName;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
    SD_BUFFER_SPEC pInfo;
} SDARG_Word_GetInfo;

typedef struct {
    SDARG_HEADER h;
    SD_WORD ret;
    SD_VOC hVocTarget;
    SD_VOC hVocSource;
    SD_WORD hWordSource;
} SDARG_Word_CopyFrom;

typedef struct {
    SDARG_HEADER h;
    SD_WORD ret;
    SD_VOC hTargetVoc;
    int16 nSourceVocs;
    int16 nSourceWords;
    int16 nUtts;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC phSourceVocs;
    SD_BUFFER_SPEC pSourceWords;
    SD_BUFFER_SPEC pUtts;
} SDARG_Word_BuildFrom;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
```

```
    SD_BUFFER_SPEC pName;
} SDARG_Word_Lookup;

typedef struct {
    SDARG_HEADER h;
    SD_USER hUserTarget;
    SD_VOC hVoc;
    SD_WORD hWord;
    SD_BUFFER_SPEC pNewName;
} SDARG_Word_CopyToUser;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    uns32 index;
    uns32 nBuf;
    SD_BUFFER_SPEC pBuf;
} SDARG_Word_List;

typedef struct {
    SDARG_HEADER h;
    SD_WORD ret;
    SD_VOC hVoc;
    SD_BUFFER_SPEC pIter;
} SDARG_Word_Iterate;

typedef struct {
    SDARG_HEADER h;
    SD_WORD ret;
    SD_BUFFER_SPEC pIter;
} SDARG_Word_Next;

typedef struct {
    SDARG_HEADER h;
    int16 ret;
    SD_VOC hVoc;
    SD_WORD hWord;
} SDARG_Word_HasModel;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_VOC hVoc;
    SD_WORD hWord;
} SDARG_Word_StateRefCount;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    uns32 nList;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pList;
    SD_BUFFER_SPEC pResults;
    SD_BUFFER_SPEC pPar;
    SD_BUFFER_SPEC pRecStatus;
    SD_BUFFER_SPEC pContext;
    SD_BUFFER_SPEC prefix;
} SDARG_Word_Recog;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    uns32 nBuf;
    int16 caseSensitive;
#ifndef _MACINTOSH_
    uns16 ds;
#else
    long A5;
#endif
    uns32 (SD_CALLBACK *fBreak)(void);
    SD_BUFFER_SPEC pBuf;
    SD_BUFFER_SPEC prefix;
) SDARG_Word_ListMatchesByFrequency;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
    uns32 nUtt;
    SD_BUFFER_SPEC pUtt;
    SD_BUFFER_SPEC pPar;
    SD_BUFFER_SPEC pStatus;
    SD_BUFFER_SPEC pRejCodes;
) SDARG_VocId_Build;

typedef struct (
    SDARG_HEADER h;
    uns16 ret;
    SD_VOC hVoc;
    SD_WORD hWord;
    SD_BUFFER_SPEC pBuf;
) SDARG_VocId_GetPronunciation;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
    SD_BUFFER_SPEC pPron;
) SDARG_VocId_SetPronunciation;

typedef struct (
    SDARG_HEADER h;
    SD_STATE ret;
    SD_VOC hVoc;
    SD_STATE hParent;
) SDARG_State_New;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
) SDARG_State_Delete;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
) SDARG_State_Load;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
) SDARG_State_Save;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
```

SDAPIX.H 7-22-95 2:09p

```
) SDARG_State_Unload;

typedef struct (
    SDARG_HEADER h;
    SD_VOC ret;
    SD_VOC hVocTarget;
    SD_VOC hParentTarget;
    SD_VOC hVocSource;
    SD_STATE hStateSource;
    uns16 copyChildren;
    uns16 renameTarget;
    SD_BUFFER_SPEC newName;
) SDARG_State_CopyFrom;

typedef struct (
    SDARG_HEADER h;
    uns16 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pBuf;
) SDARG_State_GetName;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pStateName;
) SDARG_State_SetName;

typedef struct (
    SDARG_HEADER h;
    SD_STATE ret;
    SD_VOC hVoc;
    SD_STATE hParent;
    SD_BUFFER_SPEC pStateName;
) SDARG_State_GetHandle;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pInfo;
) SDARG_State_GetInfo;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pInfo;
) SDARG_State_GetExtendedInfo;

typedef struct (
    SDARG_HEADER h;
    SD_TRANSITION ret;
    SD_VOC hVoc;
    SD_STATE hState;
) SDARG_State_GetTransition;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_TRANSITION trans;
) SDARG_State_SetTransition;

typedef struct (
```

Page 9 of 15

```
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pBuf;
} SDARG_State_GetEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pBuf;
} SDARG_State_SetEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pIter;
} SDARG_State_IterateEnv;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_BUFFER_SPEC pBuf;
    SD_BUFFER_SPEC pIter;
} SDARG_State_NextEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
} SDARG_State_AddWord;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
} SDARG_State_DeleteWord;

typedef struct {
    SDARG_HEADER h;
    int16 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
} SDARG_State_IsWordInState;

typedef struct {
    SDARG_HEADER h;
    int16 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
    int16 Active;
} SDARG_State_SetWordActive;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
```

SDAPIX.H 7-22-95 2:09p

```
    SD_BUFFER_SPEC pInfo;
} SDARG_State_GetWordInfo;

typedef struct {
    SDARG_HEADER h;
    SD_TRANSITION ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
} SDARG_State_GetWordTransition;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
    SD_TRANSITION trans;
} SDARG_State_SetWordTransition;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pBuf;
} SDARG_State_GetWordEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pBuf;
} SDARG_State_SetWordEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
    SD_BUFFER_SPEC pIter;
} SDARG_State_IterateWordEnv;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_BUFFER_SPEC pBuf;
    SD_BUFFER_SPEC pIter;
} SDARG_State_NextWordEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pIter;
} SDARG_State_IterateWords;

typedef struct {
    SDARG_HEADER h;
    SD_WORD ret;
    SD_BUFFER_SPEC pIter;
} SDARG_State_NextWord;
```

Page 10 of 15

```
typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    int16 order;
    int16 restrict;
    SD_BUFFER_SPEC pIter;
} SDARG_State_IterateSpecifiedWords;

typedef struct {
    SDARG_HEADER h;
    SD_WORD ret;
    uns32 nBuf;
    SD_BUFFER_SPEC pBuf;
    SD_BUFFER_SPEC pIter;
} SDARG_State_NextSpecifiedWords;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    int16  order;
} SDARG_State_SetWordOrder;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    int16  order;
    SD_BUFFER_SPEC pName;
} SDARG_State_LookupWord;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    int16  order;
    uns32 index;
    uns32 nBuf;
    SD_BUFFER_SPEC pBuf;
} SDARG_State_ListWords;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_WORD hWord;
    uns32 index;
} SDARG_State_SetWordIndex;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    uns32 nBuf;
    SD_BUFFER_SPEC pBuf;
} SDARG_State_AddWordList;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_STATE hSubState;
    uns32 index;
```

SDAPIX.H 7-22-95 2:09p

```
} SDARG_State_AddState;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_STATE hSubState;
} SDARG_State_DeleteState;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_STATE hSubState;
} SDARG_State_GetStateIndex;

typedef struct {
    SDARG_HEADER h;
    SD_TRANSITION ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_STATE hSubState;
} SDARG_State_GetStateTransition;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_STATE hSubState;
    SD_TRANSITION trans;
} SDARG_State_SetStateTransition;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_STATE hSubState;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pBuf;
} SDARG_State_GetStateEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_STATE hSubState;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pBuf;
} SDARG_State_SetStateEnv;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_STATE hSubState;
    SD_BUFFER_SPEC pIter;
} SDARG_State_IterateStateEnv;

typedef struct {
    SDARG_HEADER h;
    uns32 ret;
    SD_BUFFER_SPEC pBuf;
    SD_BUFFER_SPEC pIter;
} SDARG_State_NextStateEnv;
```

Page 11 of 15

```
typedef struct {
    SDARG_HEADER h;
    SD_STATE ret;
    SD_VOC hVoc;
    SD_STATE hState;
} SDARG_State_GetParent;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_STATE hParent;
} SDARG_State_SetParent;

typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pIter;
} SDARG_State_IterateStates;

typedef struct {
    SDARG_HEADER h;
    SD_STATE ret;
    SD_BUFFER_SPEC pIter;
} SDARG_State_NextState;

STRUC(SDARG_State_Recog) {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pResults;
    SD_BUFFER_SPEC pPar;
    SD_BUFFER_SPEC pRecStatus;
    SD_BUFFER_SPEC pContext;
    SD_BUFFER_SPEC prefix;
};

STRUC(SDARG_State_Recog2) {
    SDARG_HEADER h;
    uns32 ret;
    int32 nStates;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pStates;
    SD_BUFFER_SPEC pContext;
    SD_BUFFER_SPEC prefix;
    SD_BUFFER_SPEC pResults;
    SD_BUFFER_SPEC pRecStatus;
};

STRUC(SDARG_ContRecog) {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_STATE_MACHINE hStMach;
    SD_UTT hUtt;
    int16  nTranscription;
    SD_BUFFER_SPEC pResults;
    SD_BUFFER_SPEC pPar;
    SD_BUFFER_SPEC pRecStatus;
    SD_BUFFER_SPEC pContext;
    SD_BUFFER_SPEC prefix;
    SD_BUFFER_SPEC pTranscription;
```

```
};

STRUC(SDARG_ContRecog1) {                             // used to build state machine
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
        SD_STATE hState;
    SD_UTT hUtt;
    int16  nTranscription;
    int16  minWords;
    int16  maxWords;
    SD_BUFFER_SPEC pResults;
    SD_BUFFER_SPEC pRecStatus;
    SD_BUFFER_SPEC pContext;
    SD_BUFFER_SPEC prefix;
    SD_BUFFER_SPEC pTranscription;
};

STRUC(SDARG_ContRecogPartial) {
    SDARG_HEADER h;
    SD_BUFFER_SPEC pResults;
};


typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_BUFFER_SPEC pIter;
} SDARG_State_IterateChildren;

typedef struct {
    SDARG_HEADER h;
    SD_STATE ret;
    SD_BUFFER_SPEC pIter;
} SDARG_State_NextChild;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
} SDARG_Par_GetNumPar;

typedef struct {
    SDARG_HEADER h;
    uns16 ret;
    SD_BUFFER_SPEC pName;
} SDARG_Par_GetHandle;

typedef struct {
    SDARG_HEADER h;
    int16  ret;
    uns16 hPar;
    SD_BUFFER_SPEC pBuf;
} SDARG_Par_GetName;

typedef struct {
    SDARG_HEADER h;
    int16  ret;
    uns16 hPar;
} SDARG_Par_GetType;

typedef struct {
    SDARG_HEADER h;
    uns16 hPar;
    SD_BUFFER_SPEC pVal;
} SDARG_Par_GetValue;
```

```
typedef struct (
    SDARG_HEADER h;
    uns16 hPar;
    SD_BUFFER_SPEC pVal;
) SDARG_Par_SetValue;

typedef struct (
    SDARG_HEADER   h;
    SD_PAR         ret;
    SD_BUFFER_SPEC pIter;
) SDARG_Par_Iterate;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
    SD_BUFFER_SPEC pPar;
    SD_BUFFER_SPEC pContext;
) SDARG_Word_AdaptLm;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    int16 nWords;
    int16 nUtt;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pUtt;
    SD_BUFFER_SPEC pPar;
    SD_BUFFER_SPEC pStatus;
    SD_BUFFER_SPEC pRejCodes;
) SDARG_Word_Adapt;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    int16 nWords;
    int16 nUtt;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pUtt;
    SD_BUFFER_SPEC pPar;
    SD_BUFFER_SPEC pStatus;
    SD_BUFFER_SPEC pRejCodes;
) SDARG_Word_Train;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    int16 nWords;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pName;
    SD_BUFFER_SPEC pResults;
    SD_BUFFER_SPEC pRecPar;
    SD_BUFFER_SPEC pRecStatus;
    SD_BUFFER_SPEC pContext;
    SD_BUFFER_SPEC prefix;
) SDARG_Word_Learn;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
) SDARG_VocPool_New;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
) SDARG_VocPool_Delete;

typedef struct (
    SDARG_HEADER h;
    int16 ret;
    SD_VOC hVoc;
) SDARG_VocPool_Exists;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
) SDARG_VocPool_TouchId;

typedef struct (
    SDARG_HEADER h;
    SD_WORD ret;
    SD_VOC hVoc;
) SDARG_VocPool_GetNextId;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
) SDARG_VocPool_LockId;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
) SDARG_VocPool_UnlockId;

typedef struct (
    SDARG_HEADER h;
    int32 ret;
    SD_VOC hVoc;
    SD_WORD hWord;
) SDARG_VocPool_GetLockCnt;

typedef struct (
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
    int32 cnt;
) SDARG_VocPool_SetLockCnt;


typedef struct (
    SDARG_HEADER h;
    SD_STATE_MACHINE ret;
    SD_STATE hState;
) SDARG_StMach_New;

typedef struct (
    SDARG_HEADER h;
    SD_STATE_MACHINE machine;
) SDARG_StMach_Delete;

typedef struct (
    SDARG_HEADER h;
    SD_STATE_MACHINE ret;
    SD_VOC hVoc;
    SD_STATE_MACHINE hMachin;
    SD_WORD hWord;
    int16  del;
) SDARG_StMach_Update;
```

```
typedef struct {
    SDARG_HEADER h;
    SD_STATE ret;
    SD_STATE_MACHINE hStMach;
} SDARG_StMach_GetCurrentState;

/* What does a parse consist of?
 * Answer: For each word, the chain of called/included states that led to the
 * state defining the word such that the transition for the word in that state
 * is a transition in a successul parse.
 * We need to have either a disambiguation strategy or a way of returning
 * multiple candidate parses.
 */

STRUC( SDARG_StMach_Parse ) {
    SDARG_HEADER h;
    BOOL16 ret;
    SD_STATE_MACHINE hStMach;
    int16 nWords;
    int16 lParse;
    SD_BUFFER_SPEC pWordSpecs;
    SD_BUFFER_SPEC pParseBuf;
};

// 2-15-93 CEI Added STRUC(SDARG_Lm_Reorder) for right context lm
STRUC(SDARG_Lm_Reorder) {
    SDARG_HEADER h;
    int16 ret;
    int16 nStates;
    int16 nChoicesL;                // length of first buffer (left)
    int16 nChoicesR;                // length of second buffer (right)
    SD_BUFFER_SPEC pStates;
    SD_BUFFER_SPEC pResultsL;     // first choice list (left)
    SD_BUFFER_SPEC pContextL;
    SD_BUFFER_SPEC pResultsR;     // second choice list (right)
    SD_BUFFER_SPEC pContextR;
};

// for inhouse call SDLm_DecayAdjust (GRG, 1SEP93)
typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
} SDARG_Lm_DecayAdjust;

// for inhouse call SDLm_GetCats (GRG, 1SEP93)
typedef struct {
    SDARG_HEADER h;
    SD_VOC hVoc;
    int32 nWords;
    int16 posPrior;
    uns16 lBufs;
    SD_BUFFER_SPEC pWords;
    SD_BUFFER_SPEC pBufs;
} SDARG_Lm_GetCats;

#ifdef _Windows
/* from dispatch.cpp: */
#ifdef __HIGHC__
 #ifdef __WINDOWS386_H
int16  FAR PASCAL SDXDispatch(int funcid,SD_TASK hTask, void *pargs, uns16 largs
  s> );
 #endif
#else
int _export _far _pascal SDXDispatch(int funcid,SD_TASK hTask,
                                     void *pargs, uns16 largs);
```

```
#endif
#endif  /* _Windows */

#define SDERR_NOT_INSTALLED (-1)
#define SDERR_NOT_INIT (-2)

/* From error.c:
*/
void sdErrorHandler(SD_TASK hTask, int funcid, int errorCode);


/* INHOUSE argument blocks: */
STRUC( SDARG_Lm_Score ) {
    SDARG_HEADER h;
    SD_VOC hVoc;
    int32 nWords;
    SD_BUFFER_SPEC pWords;
    SD_BUFFER_SPEC pContext;
    SD_BUFFER_SPEC pScores;
};

// extended recog function has extra inhouse arguments in the parameter
// structure
STRUC( SDARG_State_Recogx ) {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_STATE hState;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pResults;
    SD_BUFFER_SPEC pPar;
    SD_BUFFER_SPEC pRecStatus;
    SD_BUFFER_SPEC pContext;
    SD_BUFFER_SPEC prefix;
    SD_BUFFER_SPEC pPassThru;
};

STRUC( SDARG_Word_ListIds ) {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_WORD hWord;
    uns32 nBuf;
    SD_BUFFER_SPEC pBuf;
};

STRUC( SDARG_Word_GetPronunciation ) {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_WORD hWord;
    SD_BUFFER_SPEC pBuf;
};

STRUC( SDARG_Word_GetLmlCount ) {
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_WORD hWord;
};

STRUC( SDARG_Word_SetLmlCount ) {
    SDARG_HEADER h;
    SD_VOC hVoc;
    SD_WORD hWord;
    uns32 lmlCount;
);

STRUC( SDARG_Word_GetlmlCountWithRecent ) (
    SDARG_HEADER h;
    uns32 ret;
    SD_VOC hVoc;
    SD_WORD hWord;
);

STRUC( SDARG_Utt_Get ) (
    SDARG_HEADER h;
    uns32 ret;
    SD_UTT hUtt;
    SD_BUFFER_SPEC pBuf;
);


STRUC( SDARG_Utt_MakeRandom ) (
    SDARG_HEADER h;
    SD_UTT ret;
    SD_WORD hWord;
    SD_VOC hVoc;
);

typedef struct (
    SDARG_HEADER h;
    uns16 ret;
    SDAPI_FUNC_CALLBACK pEntryFunc;
    SDAPI_FUNC_CALLBACK pExitFunc;
) SDARG_SetFuncCallBack;

#endif /* _SDAPIX_H_ */
```

```
/* STACK.H*/
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef stack_h
#define stack_h


#include "ordcoll.h"


////////////////////////////////////////////////////////////////////////
// Stacks are OC's with push() pop() and top() access commands...
// Stacks grow down...  top() == *last()
////////////////////////////////////////////////////////////////////////


template <class T>
class Stack : public OC< T >
{
public:
    Stack( int initialSize = 8, int incrementSize = 8 ) : OC< T >( initialSize,
  => incrementSize ) { }

    ///////////////////////////////////////////////////////////////////
    // create new inline Stack functions...
    T* push( T* v )          { OCBase::add( v );   return v; }
    T* pop()          { return( isEmpty() ? 0 : removeEntry( count() - 1 ) ); }
    T* top() const   { return( isEmpty() ? 0 : (*this)[ count() - 1 ] ); }

};


template <class T>
class PriorityQue : public OC< T >
{
 protected:
    CompareOCEntries cmpfn;
public:
    PriorityQue( CompareOCEntries cmpfunc, int initialSize = 8, int incrementSiz
  => e = 8 )
    : OC< T >( initialSize, incrementSize ) {
        cmpfn = cmpfunc;
    }

    ///////////////////////////////////////////////////////////////////
    // create new inline Stack functions...
    T* push( T* e ) { int position;  find( e, cmpfn, &position );  add( e, posit
  => ion );  return e; }
    T* pop()          { return( isEmpty() ? 0 : removeEntry( count() - 1 ) ); }
    T* top() const   { return( isEmpty() ? 0 : (*this)[ count() - 1 ] ); }
};


#endif /* stack_h */
```

```
/*
   slidectl.h : dragon horizontal slide control

   PROJECT: DragonDictate for Windows -- Custom Controls
   CREATED: February 4, 1994
   AUTHOR:  Joel Gould

   (C) Copyright Dragon Systems, Inc. 1993-1994.
   All Rights Reserved.

   ** DRAGON SYSTEMS CONFIDENTIAL **

.......................................................................

   DESCRIPTION:

   This is a slide control which is similar to a ScrollBar control.  This
   control is designed to be used in the options dialog.

   The following style bits are supported:

        SBS_HORZ
            This style should always be set when the slider is used.
            (Actually the currently implementation does not use this
            bit, but I may in future.)

        WS_VISIBLE
        WS_TABSTOP
        WS_DISABLED
        WS_GROUP
            These are the standard control styles for dialog boxes.

   The slide control will do nothing with its window caption, so the caption
   need not be set.

   The slide control is available from the Borland resource workshop
   if you install the control.dll.

   To access this control, use the scrollbar functions as if this was
   a Windows scrollbar control.  You can also directly send the window
   WM_USER messages.

        SetScrollPos( hWnd, SB_CTL, nPos, bRedraw )                 -or-
        SendWindowMessage( hWnd, WM_USER+0, nPos, bRedraw )

        nPos = GetScrollPos( hWnd, SB_CTL )                         -or-
        nPos = SendWindowMessage( hWnd, WM_USER+1, 0, 0L )

        SetScrollRange( hWnd, SB_CTL, nMin, nMax, bRedraw )         -or-
        SendWindowMessage( hWnd, WM_USER+2, bRedraw, MAKELONG( nMin, nMax ) )

        GetScrollRange( hWnd, SB_CTL, &nMin, &nMax )                -or-
        nMin = LOWORD( SendWindowMessage( hWnd, WM_USER+3, 0, 0L ) )
        nMax = HIWORD( SendWindowMessage( hWnd, WM_USER+3, 0, 0L ) )

   This control will NOT send WM_VSROLL and WM_HSCROLL messages.  Instead,
   it will send a WM_CONTROL message when its value changes and you must
   then query its value.

.......................................................................

   TO DO LIST:

.......................................................................

   MODIFICATIONS:
                              SLIDECTL.H  7-22-95  2:09p
```

```
      *tlib-revision-history*
     1 SLIDECTL.H 25-May-94,16:15:16,`STIJN' First version under TLIB
      *tlib-revision-history*

      Revision 2 on Tue Mar 01 18:25:20 1994 by STEVES
     DDWin Version 0.00.04.112
      added OnEraseBkGnd
      adopted create-GDI-objects-during-paint model for greater reliability
        and easier adaptation to changing colors

*/

#ifndef slidectl_h
#define slidectl_h



#define WM_SETSCROLLPOS   WM_USER+0
#define WM_GETSCROLLPOS   WM_USER+1
#define WM_SETSCROLLRANGE WM_USER+2
#define WM_GETSCROLLRANGE WM_USER+3



#define DGN_SLIDECONTROL "DgnSlideControl"

//////////////////////////////////////////////////////////////////////
//

class DgnSlideControl
(
public:
    // call this routine to initialize the control

    void libMainInit( );

protected:

    // this is the offset to the extra data area for this control

    int m_myOffset;

    // address of the default window prof

    WNDPROC m_myDefProc;

    // this is used in the wndproc to indicate when the message is handled

    BOOL m_bMessageHandled;

    // this utility subroutine converts a pixel position into a slide
    // position

    int ComputeOffset( HWND hWnd, int xPixel );

    // this utility routine is the oposite of ComputeOffset

    int ComputePixel( HWND hWnd, int nOffset );

    // this utility subroutine redraws the dotted rectangle with a new
    // flash position and updates the flash position

    void UpdateFlash( HWND hWnd, int nOffset );

    // this utility subroutine changes the current slide position and
    // alerts our parent
                              Page 1 of 2
    void UpdatePosition( HWND hWnd, int nPos );

    // this utility routine draws the flash ON or Off at a screen position

    void DrawFlash( HWND hWnd, HDC hDC, int nOffset, BOOL bState );

    // this utility routine computes a variety of drawing offsets

    void GetMeasurements( HWND hWnd,
                          int & totalUnits, int & halfWidth,
                          int & fullHeight, int & rangeWidth );

    // this utility recomputes and sets the tic ratio

    void UpdateTicCount( HWND hWnd );

    // we have a number of local window variables;  these routines access
    // those variables for the current window.
    //
    //      offset 0: offset which has flashing rectangle
    //      offset 2: position of scroll bar
    //      offset 4: minimum value for scrollbar
    //      offset 6: maximum value for scrollbar
    //      offset 8: number of tic marks less 1 (min: 1)

    int  GetFlashState( HWND hWnd );
    void SetFlashState( HWND hWnd, int nValue );

    int  GetScrollPos( HWND hWnd );
    void SetScrollPos( HWND hWnd, int nValue );

    int  GetScrollMin( HWND hWnd );
    void SetScrollMin( HWND hWnd, int nValue );

    int  GetScrollMax( HWND hWnd );
    void SetScrollMax( HWND hWnd, int nValue );

    int  GetTicCount( HWND hWnd );
    void SetTicCount( HWND hWnd, int nValue );

    // window messages we handle

    int  OnCreate( HWND hWnd, CREATESTRUCT *lpCreateStruct );
    void OnNCDestroy( HWND hWnd );
    void OnSetFocus( HWND hwnd, HWND hwndOldFocus );
    void OnKillFocus( HWND hwnd, HWND hwndNewFocus );
    void OnPaint( HWND hWnd );
    BOOL OnEraseBkGnd( HWND hWnd, HDC hdc );
    void OnTimer( HWND hWnd, UINT id );
    void OnSize( HWND hWnd, UINT state, int cx, int cy );
    void OnSetScrollPos( HWND hWnd, int nValue, BOOL fRedraw );
    void OnSetScrollRange( HWND hWnd, int nMin, int nMax, BOOL fRedraw );
    int  OnGetScrollPos( HWND hWnd );
    long OnGetScrollRange( HWND hWnd );
    UINT OnGetDlgCode( HWND hWnd, MSG FAR* lpmsg );
    void OnLButtonDown( HWND hWnd, BOOL fDoubleClick, int x, int y, UINT keyFlag
  => s );
    void OnLButtonUp( HWND hWnd, int x, int y, UINT keyFlags );
    void OnMouseMove( HWND hWnd, int x, int y, UINT keyFlags );
    void OnKeyDown( HWND hWnd, UINT vk, BOOL fDown, int cRepeat, UINT flags );

public:

    LRESULT wndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam );
```

SLIDECTL.H 7-22-95 2:09p

```
};

extern DgnSlideControl slideControl;

#endif
```

Page 2 of 2

```
/* WAPP.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#include <windows.h>


#define IDI_MY_ICON                1

#define USR_COMBO_ID               101
#define CHOICE_LIST_ID             102
#define EXIT_BTN_ID                103
#define MICROPHONE_BTN_ID          104
#define CONFIDENCE_TEXT_ID         105

#define REC_ZIP_TEXT_ID            115
#define REC_STATE_TEXT_ID          116
#define REC_CITY_TEXT_ID           117
#define STAT1_TEXT_ID              118
#define STAT2_TEXT_ID              119
#define STAT3_TEXT_ID              120
#define PROMPT_TEXT_ID             121
#define SAVE_BTN_ID                122
#define SETUSER_BTN_ID             123
#define VUMETER_ID                 124
#define FIX_ERRORS_BTN_ID          125
#define VUTIMER_ID                 126
#define PURGE_BTN_ID               127
#define MEMORY_TEXT_ID             128

#define WORD_PAUSE_TEXT_ID         106
#define START_THRESH_TEXT_ID       107
#define END_THRESH_TEXT_ID         108

#define WORD_PAUSE_SLIDE_ID        110
#define START_THRESH_SLIDE_ID      111
#define END_THRESH_SLIDE_ID        112


// UserMessages...

#define UM_CHANNELSTART            WM_USER+1001        // start of utterance
#define UM_TRAINWORDS              WM_USER+1002        // start test dialog box
#define UM_EXIT                    WM_USER+1003        // exit dialog box
#define UM_INIT_RECOGNIZER         WM_USER+1004        // start of utterance
#define UM_INIT_DIALOG             WM_USER+1005        // start of utterance
#define UM_DDE_TEXT_TO_SPEECH      WM_USER+1006


#define  DDE_TIMEOUT 3000
#define  MAX_SIZE  1000
#define  BUFFER_SIZE 1024*8
```

WAPP.H 7-22-95 2:12p

Page 1 of 1

```
/* $Header:   G:/usr/local/src/lib/vcs/std.h_v   1.23   16 Nov 1993 17:26:42   J
  => OSHUA  $ */
/* STD.H  / 11 Apr-86 / (C) Copyright Dragon Systems, Inc.  1985 - 1991

   Dragon standard definitions - #include this file FIRST
       (unless you intend to include WINDOWS.H, in which case
         #include WSTD.H and not STD.H)

   Modifications:
   $Log:   G:/usr/local/src/lib/vcs/std.h_v  $
 *
 *    Rev 1.23   16 Nov 1993 17:26:42   JOSHUA
 * Changes for BorlandC version 4 / 32 bits
 *
  Changes for Borland C 4 / 32 bits.  Also, please
  define 16 bits for new 16 bit compilers (to aid the transition
  to 64 bits one day!)
 *
 *    Rev 1.22   03 Aug 1993 15:17:50   DEAN
 * define _SIZE_T_DEFINED for Microsoft and HighC
 *
 *    Rev 1.21   20 Jul 1993 16:18:58   JOSHUA
 * #undef near, because watcom defines it
 *
 *    Rev 1.20   16 Jul 1993 14:38:08   BETSY
   Additional defines for the Macintosh.
 *
 *    Rev 1.19   24 Jun 1993 10:08:40   DEAN
 * Oops - this time for sure!
 *
 *    Rev 1.18   23 Jun 1993 15:16:48   DEAN
   Paulvm: #undef far and huge before #defining them (for Watcom's benefit).
 *
 *    Rev 1.17   23 Jun 1993 11:54:16   BETSY
   Mac types. Removed DurGetA5.
 *
 *    Rev 1.16   09 Jun 1993 14:56:40   BETSY
 * for Driver version 4.11.52.
   Macintosh def of XFAR. Moved INC_PTR here.
 *
 *    Rev 1.15   01 Jun 1993 15:00:12   DEAN
 * ZORTECH support
 *
 *    Rev 1.14   16 Mar 1993 16:19:38   BETSY
 * Added ifdefs for _MAC
 *    Added define for MAC if MPW or Think C compiler.
 *    Defined _32BIT_  on for MAC.
 *    Added macro for icrementing a pointer by the size of a struct.
 *
 *    Rev 1.13   10 Mar 1993 10:54:18   DEAN
 * cleaned up 32-bit defs
 *
 *    Rev 1.12   02 Jan 1993 11:36:52   DEAN
   ifdef _32BIT_ define huge to be the empty string.
 *
 *    Rev 1.11   31 Dec 1992 15:02:18   DEAN
   ifdef _32BIT_ define far to be the empty string.
 *
 *    Rev 1.10   04 Mar 1992 10:07:42   DEAN
 * defined LOW_16 and HIGH_16 and MAKE_32.
   ifndef _I86, we were defining _FAR instead of XFAR.
 *
 *    Rev 1.9   11 Feb 1992 16:53:42   PETER
 * Change the definition of VOID to be a #define instead of a typedef.  This
 * makes std.h compatible with windows.h now.
 *
```

```
 *    Rev 1.8   30 Jan 1992 13:34:06   JOEL
 * Enclosed #define NULL inside #ifndef NULL  .... #endif
 *
 *    Rev 1.7   10 Jan 1992 09:42:42   DEAN
 * as per Peter's suggestion, define NDEBUG ifndef DEBUG
 *
 *    Rev 1.6   12 Dec 1991 14:25:46   DEAN
 * replaced C++-style comments with C-style comments.
 *
 *    Rev 1.5   15 Nov 1991 08:20:36   DEAN
 * OOPS! _BIGGEST_SIZE_ was under the conditional #ifdef _32BIT instead of
 * _32BIT_. This affects the proper compilation of MEM.C in the library.
 *
 *    Rev 1.4   14 Oct 1991 13:31:32   DEAN
 * Cosmetic change - don't #define BOOL_DEFINED if it isn't already defined.
 *
 *    Rev 1.3   10 Oct 1991 15:57:58   DEAN
 * This time for sure.
 *
 *    Rev 1.1   10 Oct 1991 15:06:32   DEAN
 * BUG FIX in case wstd.h is used

    10-Oct-91 DGS
    BUG FIX. Need to define BYTE if BYTE_DEFINED is defined.

 *
 *    Rev 1.0   09 Oct 1991 11:06:06   DEAN
 * Initial revision.

    8-Oct-91 DGS
    Major rewrite as a result of software standards meeting.
*/

#ifndef _STD_H_  /* protect against multiple includes of this file */
#define _STD_H_


/* Supported compilers and definitions, some automatically defined, some not

__TURBOC                 Turbo OR Borland C compiler (automatically defined)
__BORLANDC__             Borland C compiler (automatically defined),
                         use only if using some feature of Borland C not
                         shared by Turbo C.
__HIGHC__                Metaware High C Compiler   (automatically defined)
_MSC                     Microsoft C compiler (NOT automatically defined)
__WATCOMC__              Watcom C compiler (is this auto-defined?)

_UNIX_                   RS6000 compiler (NOT automatically defined)

_MACINTOSH_              defined if MPW or Think C
applec                   MPW Macintosh C compiler ( automatically defined )
THINK_C                  Think C compiler for Mac ( automatically defined )

*/

#ifdef _MSC
#define __MSDOS__          /* Microsoft C defines MSDOS */
                           /* FUTURE: define 16BIT/32BIT as appropriate */
#endif

#ifdef __BORLANDC
#ifdef __WIN32__  7* Borland C 4 defines this for 32 bit apps */
                  /* (including console apps) */
 #define _32BIT_
#else
 #define _16BIT_
```

```
#endif
#endif

#ifdef __HIGHC__
 #define _32BIT_
 #define __MSDOS__
#endif

#ifdef __ZTC__          /* Zortech */
 #define _32BIT_
 #define __MSDOS__
#endif

#ifdef __WATCOMC__
 #define __MSDOS__
 #define _32BIT_
#endif

#ifdef _UNIX_
 #define _BIG_ENDIAN_
 #define _32BIT_
#endif

#if  defined(applec) || defined(THINK_C) || defined(__SC__)
 #define _MACINTOSH_
 #define _32BIT_
 #define _BIG_ENDIAN_
#endif

#ifdef __MSDOS__
 #define _I86_
#endif

/* Constants
*/
#define YES     1
#define NO      0
#define TRUE    1
#define FALSE   0

/* far pointer definitions */
#if defined(_I86_) || defined(_MACINTOSH_)
 #ifdef _32BIT_
        #define HUGE                    /* all pointers are HUGE! */
        #define FAR                     /* all pointers are far! */
        #undef far          /* Watcom compiler #defines this! */
        #define far
        #undef huge         /* Watcom compiler #defines this! */
        #define huge
        #undef near         /* Watcom compiler #defines this! */
        #define near
        #ifndef _MACINTOSH_
                #define XFAR _far       /* I don't know whether all 32-bit compi
  => lers
                                support this definition of a 48-bit pointer,
                                but HighC does */
                #else
                #define XFAR
                #endif
          #define _NEAR
          #define _FAR
 #else  /* _32BIT_ not defined */
          #define HUGE huge
          #define FAR _far
          #define XFAR far      /* in 16-bit, only need seg:off */
 #endif /* _32BIT_ */
```

STD.H 7-22-95 2:09p

```
#else /* _I86_ not defined */
 #define HUGE
 #define FAR
 #define XFAR
#endif

#ifdef __HIGHC__
 #define INT_HANDLER _far void
#else
 #define INT_HANDLER void interrupt
#endif

/* NULL Pointers
*/
#ifndef NULL
#define NULL ((void *)0)
#endif
#define HUGE_NULL ((void HUGE *)0)

/* Logical Constructs
*/
#define OR      ||
#define AND     &&
#define NOT     !
#define IS      ==
#define ISNOT   !=
#define ISNT    !=

/* Other useful Constructs
*/
#define FOREVER for(;;)

/* Universal types
*/

#ifndef BYTE_DEFINED            /* ustd.h */
 typedef unsigned char  BYTE;
#else
 #define BYTE unsigned char     /* so our definitions work */
#endif

typedef signed char      int8;
typedef short int        int16;
typedef long             int32;

typedef unsigned int     uint,UINT;
typedef unsigned char    uns8,uint8;
typedef unsigned short   uns16,uint16;
typedef unsigned long    uns32,uint32;

#ifndef _MACINTOSH_
#ifndef BOOL_DEFINED                    /* ustd.h defines this */
 typedef int BOOL;                      /* 0, non-0 type */
#endif
typedef int             bool,boolean;
#else
#include "Types.h"
typedef Boolean BOOL;
typedef Boolean bool, boolean;
#endif

typedef int8            BOOL8;
typedef int16           BOOL16;
typedef int32           BOOL32;
```

Page 2 of 3

```
#if defined( _MACINTOSH_ ) && ( defined( __SC__ ) || defined(THINK_C))
#include <size_t.h>
#else
typedef unsigned int     size_t;
#endif

#define _SIZE_T_DEFINED_     /* FOR WATCOM COMPILER */
#define _size_t              /* FOR INTEL COMPILER */
#define _SIZE_T              /* FOR BORLAND COMPILER */
#define _SIZE_T_DEFINED      /* FOR HIGHC AND MICROSOFT COMPILERS */

#define VOID                 void

typedef void *               VOID_PTR;
typedef void *               VPTR;
typedef BYTE *               BPTR;            /* byte pointer */
typedef BYTE *               BYTE_PTR;        /* alternative */

typedef char HUGE *          HBPTR;           /* huge byte pointer */
typedef char HUGE *          HUGE_BYTE_PTR;   /* alternative */
typedef void HUGE *          HVPTR;           /* huge void pointer */
typedef void HUGE *          HUGE_PTR;        /* alternative */

/* MACROS normally found in stdio.h
*/
#define ABS(x) ((x)<0?-(x):(x))
#define MAX(a,b) ((a)>(b)?(a):(b))
#define MIN(a,b) ((a)<=(b)?(a):(b))

/* Debugging features
*/
#ifdef DEBUG
#define PRIVATE
#else
#define NDEBUG               /* for ANSI standard ASSERT.H */
#define PRIVATE static
#endif
#define PUBLIC

/* number of entries in an initialized array */
#define NENTRIES(a) (sizeof(a)/sizeof((a)[0]))

/* largest size_t */
#ifdef _32BIT
#define _BIGGEST_SIZE_ (0xffffffff)
#else
#define _BIGGEST_SIZE_ (0xffff)
#endif

// assumes p is a byte pointer:
#define INC_PTR(p,t) ((p)+=sizeof(t))

/* helpful macro for defining a structure */
#define STRUCT(x) typedef struct _ ## x x; struct _ ## x
    /* structures defined using this macro can be referred to
        without the 'struct' qualifier */

/* packed structure macros */
#ifdef __HIGHC__
    #define PACKED _packed
#else
    #define PACKED
#endif
```

STD.H 7-22-95 2:09p

```
/* I'm not sure this works ... */
#define PACKED_STRUCT(x) typedef struct _ ## x x; PACKED struct _ ## x

#ifdef __cplusplus
#define EXTERN extern "C"
#else
#define EXTERN extern
#endif


#define LOW_16(u32) ((uns16)(u32))
#define HIGH_16(u32) ((uns16)((uns32)(u32)>>16))
#define MAKE_32(lo16,hi16) (((uns16)(lo16)) | (((uns32)((uns16)(hi16))) << 16))


/* compilation-stopping macros */
#ifdef _32_BIT_ONLY_
 #ifndef _32BIT_
Error! This code is meant for 32 bit compilers only.
 #endif
#endif

#ifdef _MSDOS_ONLY_
 #ifndef __MSDOS__
Error! This code is meant for MSDOS only.
 #endif
#endif

#ifdef _I86_ONLY_
 #ifndef _I86_
Error! This code is meant for Intel x86 targets only.
 #endif
#endif

/* Definitions needed for backwards compatibility - DO NOT USE!!! */
#ifdef __HIGHC__
#ifndef _HIGHC_
 #define _HIGHC_
#endif
#endif

#ifdef _BIG_ENDIAN
 #define BIG_ENDIAN
#endif

#ifndef _32BIT
 #ifndef _BIG_ENDIAN_
    #define SEGOF(p) (((uns16 *)&(p))[1])
    #define OFFOF(p) (((uns16 *)&(p))[0])
 #endif
#endif

typedef BYTE *PTR;

#ifdef BYTE      /* undo our def if BYTE_DEFINED is defined */
 #undef BYTE
#endif


#endif /* _STD_H_ not already defined */
```

Page 3 of 3

```
/* WCOACH.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#include <windows.h>


#define IDI_MY_ICON              1

#define USR_COMBO_ID             101
#define CHOICE_LIST_ID           104
#define EXIT_BTN_ID              105
#define MICROPHONE_BTN_ID        106
#define CONFIDENCE_TEXT_ID       107

#define REC_ZIP_TEXT_ID          115
#define REC_STATE_TEXT_ID        116
#define REC_CITY_TEXT_ID         117
#define STAT1_TEXT_ID            118
#define STAT2_TEXT_ID            119
#define STAT3_TEXT_ID            120
#define PROMPT_TEXT_ID           121
#define SAVE_BTN_ID              122
#define SETUSER_BTN_ID           123
#define VUMETER_ID               124
#define FIX_ERRORS_BTN_ID        125
#define VUTIMER_ID               126


#define WORD_PAUSE_SLIDE_ID      110

// UserMessages...

#define UM_CHANNELSTART          WM_USER+1001       // start of utterance
#define UM_TRAINWORDS            WM_USER+1002       // start test dialog box
#define UM_EXIT                  WM_USER+1003       // exit dialog box
#define UM_INIT_RECOGNIZER       WM_USER+1004       // start of utterance
#define UM_INIT_DIALOG           WM_USER+1005       // start of utterance
#define UM_DDE_TEXT_TO_SPEECH    WM_USER+1006


#define  DDE_TIMEOUT 3000
#define  MAX_SIZE  1000
#define  BUFFER_SIZE 1024*8
```

WCOACH.H 7-22-95 2:12p

Page 1 of 1

```
/* WENROLL.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#include <windows.h>

#define IDI_MY_ICON              1

#define USR_COMBO_ID             101
#define PROMPT_COMBO_ID          102
#define CHOICE_LIST_ID           104
#define EXIT_BTN_ID              105
#define MICROPHONE_BTN_ID        106
#define CONFIDENCE_TEXT_ID       107
#define SAVEEXIT_BTN_ID          108
#define BACKUP_BTN_ID            109

#define SAY_STATE_TXT_ID         110
#define SAY_ZIP_TXT_ID           111
#define SAY_CITY_TXT_ID          112
#define PROMPT_CITY_TXT_ID       113
#define PROMPT_STATE_TXT_ID      114
#define PROMPT_ZIP_TXT_ID        115
#define WORD_PAUSE_SLIDE_ID      116

#define ENROL_USR_COMBO_ID       120
#define ENROL_BTN_ID             121
#define EDIT_SSNUM_ID            122
#define EDIT_SHIFT_ID            123
#define RADIO_MALE_ID            124
#define RADIO_FEMALE_ID          125
#define CREATING_USER_TEXT_ID    126
#define ENROL_EXIT_BTN_ID        127


// UserMessages...

#define UM_CHANNELSTART          WM_USER+1001        // start of utterance
#define UM_TRAINWORDS            WM_USER+1002        // start test dialog box
#define UM_EXIT                  WM_USER+1003        // exit dialog box
#define UM_INIT_RECOGNIZER       WM_USER+1004        // start of utterance
#define UM_INIT_DIALOG           WM_USER+1005        // start of utterance
```

```
/*WTRAIN.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#include <windows.h>

#define IDI_MY_ICON             1

#define USR_COMBO_ID            101
#define PROMPT_COMBO_ID         102
#define CHOICE_LIST_ID          104
#define EXIT_BTN_ID             105
#define MICROPHONE_BTN_ID       106
#define CONFIDENCE_TEXT_ID      107
#define SAVEEXIT_BTN_ID         108
#define BACKUP_BTN_ID           109

#define SAY_STATE_TXT_ID        110
#define SAY_ZIP_TXT_ID          111
#define SAY_CITY_TXT_ID         112
#define PROMPT_CITY_TXT_ID      113
#define PROMPT_STATE_TXT_ID     114
#define PROMPT_ZIP_TXT_ID       115
#define WORD_PAUSE_SLIDE_ID     116

#define VUMETER_ID              125
#define VUTIMER_ID              126


// UserMessages...

#define UM_CHANNELSTART         WM_USER+1001        // start of utterance
#define UM_TRAINWORDS           WM_USER+1002        // start test dialog box
#define UM_EXIT                 WM_USER+1003        // exit dialog box
#define UM_INIT_RECOGNIZER      WM_USER+1004        // start of utterance
#define UM_INIT_DIALOG          WM_USER+1005        // start of utterance
```

```
/* ORDCOLL.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef ordcoll_h
#define ordcoll_h


#include <assert.h>
#include <stdlib.h>
#include "defs.h"


/////////////////////////////////////////////////////////////////////
//  The OCBase class is a dynamic array of pointers.
//  Before the first pointer is a 0 pointer, after the last pointer
//  is another 0 pointer.  The "ordered" part of the class means that
//  there is an indexing defined on the elements... it has nothing to
//  do about whether that indexing corresponds to a sorted order...
//  (Note: a sorting order can be maintained by defining a CompareOCEntries()
//  function and then by doing all adds at the proper indexes using
//  the findSortedPosition() function.)
/////////////////////////////////////////////////////////////////////


typedef int (*CompareOCEntries)( const void* given, const void* test );
// (given - test) ...

typedef int (*SortCompareOCEntries)( const void** given, const void** test );
// (given - test) ...


class OCBase
{
public:
    // constructors
    OCBase( int initialSize = 8, int incrementSize = 8 );
    // destructor
    ~OCBase();

    int count() const;
    bool isEmpty() const;

    int increment() const;
    int increment( int i );

    void** first() const;
    void** last() const;
    void** next( void** p ) const;
    void** prev( void** p ) const;

    void*& operator[]( const int i ) const;

    void add( void* v );
    void add( void*, int );

    void* remove( void* v );
    void* removeEntry( int i );
    void removeAll();

    int find( const void* v ) const;
    int find( const void* v, CompareOCEntries cmp, int sortedList=0 ) const;
    int find( const void* v, CompareOCEntries cmp, int* position ) const;
        // returns -1 if it cannot find the entry
        // if only a cmp function is given it is assumed it is an unsorted
        // list... we do a stupid exhaustive search
        // if the compare function is given, and sortedList is TRUE we do
```

ORDCOLL.H 7-22-95 2:12p

```
        // a log2 search
        // if a position is given we do a log2 search and
        // if position is given and the entry is not found, position will
        // be the proper position this item should be added at;

    void sort( SortCompareOCEntries cmp );

//  void append( OCBase *given );

protected:
    int arraySize;
    int arrayUsed;
    int incrementSize;
    void** array;
};



inline OCBase::~OCBase()
{
    delete [] (array -1);
}

inline int OCBase::count() const
{
    return arrayUsed;
}

inline bool OCBase::isEmpty() const
{
    return( count() == 0 );
}

inline int OCBase::increment() const
{
    return incrementSize;
}

inline int OCBase::increment( int i )
{
    int rtn = incrementSize;

    incrementSize = i;

    return rtn;
}

inline void** OCBase::first() const
{
    return array;
}

inline void** OCBase::last() const
{
    return array+arrayUsed-1;
}

inline void** OCBase::next( void** p ) const
{
    assert( p >= array );
    assert( p < array+arrayUsed );

    return( p + 1 );
}
```

Page 1 of 2

```
inline void** OCBase::prev( void** p ) const
{
    assert( p < array+arrayUsed );
    assert( p >= array);

    return( p - 1 );
}

inline void*& OCBase::operator[]( const int i ) const
{
    assert( i <= arrayUsed );

    return array[i];
}

inline void OCBase::removeAll()
{
    arrayUsed = 0;

    array[ 0 ] = 0;
}

inline void OCBase::sort( SortCompareOCEntries compare )
{
    if( count() < 2 ) return;

    qsort( array, arrayUsed, sizeof( void* ), (CompareOCEntries)compare );
}


template <class T>
class OC : public OCBase
{
public:
    OC( int initialsize = 8, int incrementsize = 8 ) : OCBase( initialsize, incr
  => ementsize ) { }

    T** first() const { return (T** ) OCBase::first(); }
    T** last() const { return (T** ) OCBase::last(); }
    T** next( T** e ) const { return (T** ) OCBase::next( (void** )e ); }
    T** prev( T** e ) const { return (T** ) OCBase::prev( (void** )e ); }

    T*& operator[]( int i ) const { return (T*& ) OCBase::operator[] (i); }
//  const T* operator[]( int i ) const { return (const T* ) OCBase::operator[](i
  => ); }

    void add( T* e ) { OCBase::add( (void* )e ); }
    void add( T* e, int i ) { OCBase::add( (void* )e, i ); }

    T* remove( T* e ) { return (T*) OCBase::remove( (void* ) e ); }
    T* removeEntry( int i ) { return (T*) OCBase::removeEntry( i ); }

    int find( T* e ) const { return OCBase::find( (void* ) e); }
    int find( T* e, CompareOCEntries cmp, int sortedList=0 ) const {
        return OCBase::find( (const void* ) e, cmp, sortedList );
    }
    int find( T* e, CompareOCEntries cmp, int* position ) const {
        return OCBase::find( (const void* ) e, cmp, position );
    }
//  void append( OC<T> *given ) { OCBase::merge( (OCBase*) given ); }
};


#endif /* ordcoll_h */
```

ORDCOLL.H 7-22-95 2:12p

Page 2 of 2

```
/* DEFS.H */
/* Copyright (c) 1995 by Dragon Systems, Inc.*/

#ifndef defs_h
#define defs_h

#include <limits.h>


typedef int bool;

#define TRUE        1
#define FALSE       0


#if !DEBUG
#define NDEBUG
#endif

#endif // defs_h
```

## APPENDIX B

```
// Hypotheses are generated in this module.


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <assert.h>
#include <ctype.h>
#include "trie.h"

#include "hashalfa.h"                   // includes NUMITEMS definition

static StringHS uniqueChoiceHypo;
static StringHS specialHypo;


static FILE *matrixFile;
static FILE *specialFile;

int HypoPO::checkStack()
{
        for( int i = 0; i< count(); i++ )
        {
                Hypothesis* hypo = (*this)[ i ];

                assert( hypo->nextDataIndex  <= hypo->dataLength );
                assert( hypo->nextWordIndex <= 100*hypo->dataLength );
        }
}

int StringHS::compare( const SLinkBase* slb1, const SLinkBase* slb2 ) const
{
        return( strcmp( ((StringSL* )slb1)->s, ((StringSL* )slb2)->s ) );
}

unsigned StringHS::keyHash( const SLinkBase* slb ) const
{
        return stringHash( ((StringSL* )slb)->s );
};

////////////////////////////////////////////////////////////////////////
// Hypothesis Stuff
////////////////////////////////////////////////////////////////////////
int* Hypothesis::permutationScore[ NUMITEMS ];
int* Hypothesis::insertionScore[ NUMITEMS ];
int* Hypothesis::deletionScore[ NUMITEMS ];

char* Hypothesis::data;
int Hypothesis::dataLength;

int Hypothesis::l2i( int l )
{
        // it is illegal to call this function except with letter
        // 'a'...'z' '0'...'9' and ' '

        // punctuation is following the letters and digits
        if( l >= ',' && l <= '.' )
                return( l - ',' + 37 );

        // digits are following the letters and sil in the table
        if( l >= '0' && l <= '9' )
                return( l - '0' + 27 );

        // it is assumed all letterArgs are lowercase...
        else
                        HYPO.CPP  10-20-95  3:11p
```

```
                return( l == ' '  ?  0  :  l - 'a' + 1 );
}

int Hypothesis::i2l( int l )
{
        // it is illegal to call this function except with letter
        // 'a'...'z' '0'...'9' and ' '

        // punctuation is following the letters and digits
        if( l >= 37 && l <= 40 )
                return( l - 37 + ',' );

        // digits are following the letters and sil in the table
        if( l >= 1 && l <= 26 )
                return( l + 'a' -1 );

        // silence and the digits
        else
                return( l ==  0  ?  ' '  :  l - 27 + '0' );
}



Hypothesis::Hypothesis()
{
        prevOperation = PERMUTATION;
        curScore = 0;
        nextWordIndex = 0;
        nextDataIndex = 0;
}


Hypothesis::Hypothesis(Hypothesis& h)
{
    memcpy( (void* )this, (void* )&h, sizeof( Hypothesis ) );
}


//

int AlphaHypothesis::initFromPhrase()
{
        // assign memory array
    char *ph = phrase;
    char *spl = spelling;

        // copy phrase into spelling without the spaces
        for( int len = 0; *ph != '\0'; )
        {
                if( isalpha( *ph ) )
                {
                        *spl++ = *ph++;
                        ++len;
                }
                else if( isdigit( *ph ) )
                {
                        *spl++ = *ph++;
                        ++len;
                }

                else if( isspace( *ph ) )
                        ++ph;

                else
                {
                        if( spl != spelling )          *spl++ = ' ';
                              Page 1 of 7
```

HYPO.CPP 10-20-95 3:11p

```
                        while( (*spl++ = *ph++) != '\0' )
                            ;
                        break;
                }
        }
        *spl = '\0';

        return( len );
}

int isPunct( char spl )
{
        if( spl == '-'
                || spl == '+'
                || spl == ',' )
                return 1;

        return 0;
}

int AlphaHypothesis::initFromSpelling()
{
    char *ph = phrase;
    char *spl = spelling;

        for( int len = 0; *spl != '\0'; )
        {
                if( isalpha( *spl )
                        || isdigit( *spl )
                        || isPunct( *spl ) )

                {
                        *ph++ = *spl++;
                        *ph++ = ' ';

                        ++len;
                }
                else
                        break;
        }

        // if there is more text in the spelling buffer
        while( isspace( *spl ) )
                ++spl;

        if( *spl )
        {
                ++len;
                while( *spl )
                        *ph++ = *spl++;
        }

        *ph = '\0';

        return( len );
}

//////////////////////////////////////////////////////////////////////////////
// AlphaHypoAC member function
//////////////////////////////////////////////////////////////////////////////

void AlphaHypoAC::cleanUpNewWords( Recognizer* recog )
{
    for( int i=count(); i--; )
```

Page 2 of 7

```
    {
        if( (*this)[ i ].isNewWord )
        {
            recog->deleteWord( (*this)[ i ].wordId );

            (*this)[ i ].isNewWord = 0;
        }
    }
}

// constructor
AlphaHypothesis::AlphaHypothesis( AlphaHypothesis& zr )
{
    memcpy( (void* )this, (void* )&zr, sizeof( AlphaHypothesis ) );
}

// constructor
AlphaHypothesis::AlphaHypothesis( Recognizer* recog, AlphaHypothesis& zr )
{
    memcpy( (void* )this, (void* )&zr, sizeof( AlphaHypothesis ) );

        isNewWord = ( (wordId=recog->wordId( spelling )) == 0 );

    if( isNewWord )
        wordId = recog->buildWord( spelling, phrase );

    else
        recog->addWord( wordId );

    assert( wordId != 0 );
}


// constructor
// 3 parameters: recognizer , the zip and the ZipResult class

AlphaHypothesis::AlphaHypothesis( Recognizer* recog, char *letter, AlphaHypothes
  => is& zr )
{
        // set boolean to 1 if the phrase does not have a word ID yet.
        // If this word does not exists yet, most often this is the case,
        // create a new model and add it to the state. Look in dragcpp.cpp
        // for buildWord() and addWord()

    memcpy( (void* )this, (void* )&zr, sizeof( AlphaHypothesis ) );

        strcpy( spelling, letter);

        // set phrase correct
        int numWords = initFromSpelling();

        phrase[ numWords*2 - 1 ] = '\0';

    isNewWord = ( (wordId=recog->wordId( spelling )) == 0 );

        // buildWord builds and adds the WORD_ID to the state
    if( isNewWord )
        wordId = recog->buildWord( spelling, phrase );

    else
        recog->addWord( wordId );
    assert( wordId != 0 );
}

//////////////////////////////////////////////////////////////////////////////
```

```
// Hypothesis member functions/data
//////////////////////////////////////////////////////////////////////

int compareHypothesis( const void* given, const void* test )
{
    return ((Hypothesis* )test)->curScore - ((Hypothesis* )given)->curScore;
}

OneShotHeap Hypothesis::heap( 4096 );

// get  Probability from matrix
int& Hypothesis::counts( int type, char contextL, char mainL, char emittedL )
{
        if( type == DELETION )
        return deletionScore[ l2i(mainL)][ l2i( contextL ) * 1 + l2i( emittedL )
  =>  ] ;

        else if( type == PERMUTATION )
        return permutationScore[ l2i(mainL)][ l2i( contextL ) * NUMITEMS + l2i(
  => emittedL ) ] ;

        return insertionScore[ l2i(mainL)][ l2i( contextL ) * NUMITEMS + l2i( em
  => ittedL ) ] ;
}


void Hypothesis::propagate( PriorityQue<Hypothesis>* hpq )
{

        int flag = 0;

        int curDataIndex = nextDataIndex;
        int curWordIndex = nextWordIndex;
        int prevOp = prevOperation;
        int newScore, oldScore = curScore;

        assert( curDataIndex <= dataLength );
        assert( curWordIndex <= 10*dataLength );

        wordHistory[ nextWordIndex ] = '\0';

        char history[ MAX_WORD_HISTORY ];
        strncpy( history, wordHistory, curWordIndex );
        history[ curWordIndex + 1 ] = '\0';

        int c = ( curDataIndex != 0  ?  data[ curDataIndex - 1 ]  :  ' ' );
        int context = l2i( c );

        assert( c >= 0 && c < 256 );
        assert( context >= 0 && context < 37 );

        c = data[ curDataIndex ];
        int base = l2i( c );

        assert( c >= 0 && c < 256 );
        assert( base >= 0 && base < 37 );

        {
                // do the permutation propagation
                for( int i=1; i<NUMITEMS; ++i )                 // i==emissions,
  =>  skip sil==0
                {
                        history[ curWordIndex ] = i2l( i );

                        // language modeling...
                        if( !WordDC::doWordsExist( history, curWordIndex+1 ) )
```

HYPO.CPP 10-20-95 3:11p

```
                                continue;

                        // test score for this candidate
                        if( permutationScore[ base ][ NUMITEMS*context + i ] > 7
=> 0 )
                                continue;

                        newScore = ( oldScore
                                                + permutationScore[ base ][ NUM
=> ITEMS*context + i ] );

                        // make new hypothesis if good enough score
                        // The threshold is relative to the number of letters sa
=> id.
                        // value 92 is value when no data were found in the trai
=> ning yet
                        // so, for now we consider those "unknowns" as well.

                        if( newScore < 100 )
                        {
                                Hypothesis *hypo = ( flag == 0 ?
                                                                        this :
                                                                        new( &h
=> eap ) Hypothesis( *this ) );

                                flag = 1;

                                hypo->prevOperation = PERMUTATION;

                                hypo->nextWordIndex = curWordIndex +1;
                                hypo->nextDataIndex = curDataIndex +1;

                                hypo->wordHistory[ curWordIndex ] = history[ cur
=> WordIndex ];

                                hypo->wordHistory[ curWordIndex + 1 ] = '\0';

                                hypo->curScore = newScore;


                                hpq->push( hypo );
                        }
                }
        }

        if( prevOp != INSERTION && (curWordIndex != 0 || curDataIndex+1 < dataLe
=> ngth) )
        {
                // try a deletion

                // any existing string was checked against the language model be
=> fore
                // propagating it

                history[ curWordIndex ] = '\0';

                // test score for this candidate
                if( deletionScore[ base ][ context ] < 70 )
                {
                        newScore = oldScore +  deletionScore[ base ][ context ];

                        // make new hypothesis if good enough score
                        if( newScore < 100 )
                        {
                                Hypothesis *hypo = ( flag == 0 ?
                                                                        this :
                                                                        new( &h
```

Page 3 of 7

HYPO.CPP 10-20-95 3:11p

```
=> eap ) Hypothesis( *this ) );

                        flag = 1;

                        hypo->curScore = newScore ;

                        hypo->nextWordIndex = curWordIndex;
                        hypo->nextDataIndex = curDataIndex +1;

                        hypo->wordHistory[ curWordIndex ] = '\0';

                        hypo->prevOperation = DELETION;

                        hpq->push( hypo );
                    }
                }
        }

        if( prevOp != DELETION && curWordIndex < MAX_WORD_HISTORY-dataLength-2 )
        {
                // try insertion

                for( int i=1; i<NUMITENS; ++i )                    // i==emissions,
=>  skip sil==0
                {
                        history[ curWordIndex ] = i2l( i );

                        // language modeling...
                        if( !WordDC::doWordsExist( history, curWordIndex+1 ) )
                                continue;

                        // test score for this candidate
                        if( insertionScore[ base ][ NUMITENS*context + i ] > 70
=> )
                                continue;

                        // 40 is an insertion penalty
                        newScore = ( oldScore
                                                + insertionScore[ base ][ NUMIT
=> ENS*context + i ] );

                        // make new hypothesis if good enough score
                        // The threshold is relative to the number of letters sa
=> id.
                        // value 92 is value when no data were found in the trai
=> ning yet
                        // so, for now we consider those "unknowns" as well.

                        if( newScore < 100 )
                        {
                                Hypothesis *hypo = ( flag == 0
                                                                        ? this
                                                                        : new(
=> &heap ) Hypothesis( *this ) );

                                flag = 1;

                                hypo->wordHistory[ curWordIndex ] = history[ cur
=> WordIndex ];

                                hypo->wordHistory[ curWordIndex + 1 ] = '\0';

                                hypo->nextWordIndex = curWordIndex + 1;
                                hypo->nextDataIndex = curDataIndex;

                                hypo->prevOperation = INSERTION;
```

Page 4 of 7

```
                                hypo->curScore = newScore;

                                hpq->push( hypo );
                        }
                }
        }

        // try special case
        if( curWordIndex != 0 || curDataIndex+2 < dataLength )
        {
                char pattern[ 5 ];
                pattern[ 0 ] = data[ curDataIndex ];
                pattern[ 1 ] = data[ curDataIndex + 1 ];
                pattern[ 2 ] = '\0';

                StringSL test( pattern );

                StringSL *found = specialHypo.find( &test );

                if( found != 0 )
                {
                        char emission[10];

                        strcpy( emission, found->emission );

                        int len = strlen( emission );

                        for( int m = 0; m < len; m++ )
                        {
                                history[ curWordIndex++ ] = emission[ m ];
                        }

                        history[ curWordIndex ] = '\0';

                        newScore = oldScore + 10;

                        if( newScore < 150 )
                        {
                                Hypothesis *hypo = ( flag == 0
                                                                        ? this
                                                                        : new(
=> &heap ) Hypothesis( *this ) );

                                flag = 1;


                                strcpy( hypo->wordHistory, history );

                                hypo->nextWordIndex = curWordIndex;
                                hypo->nextDataIndex += 1;

                                hypo->prevOperation = BLOCKPERMUTATION;

                                hypo->curScore = newScore;

                                hpq->push( hypo );
                        }
                }
        }
}

// Function called from within onSpeech() in wapp.cpp
// Stack decoder algoritm

// numWords corresponds to wasGoodZipPhrase in the onSpeech function
// AlphaHypoAC is a growable array. It inherits from AlphaHypothesis

int Hypothesis::build( Recognizer* recog, AlphaHypoAC* choiceList,
                       AlphaHypothesis& zr, int numWords, int numHypo  )
{
//    static PriorityQue< Hypothesis > topList( compareHypothesis );
//    static PriorityQue< Hypothesis > hypoPQ( compareHypothesis );

        // for Debugging purpose. HypoPQ inherits from PriorityQue
        // and has some additional checks
        static HypoPQ topList( compareHypothesis );
    static HypoPQ hypoPQ( compareHypothesis );
    static StringHS uniqueHypo;

    topList.removeAll();
    hypoPQ.removeAll();
    Hypothesis::heap.empty();
        uniqueHypo.removeAll();
        uniqueChoiceHypo.removeAll(); // used in propagate

    // the class Hypothesis is used to generate the hypotheses. Once
    // we have those, we will have to convert the information to
    // a AlphaHypothesis class, which contains info about the recognizer's
    // opinion on the hypothezised letter phrases

    Hypothesis *hypo = new( &Hypothesis::heap ) Hypothesis;

        Hypothesis::data = zr.spelling;
        Hypothesis::dataLength = numWords;


        hypoPQ.push( hypo );


        int count = 0;

        while( topList.count() < numHypo  &&  0 != (hypo=hypoPQ.pop()) )
        {
                // !!!!! revisit this condition for insertion and deletion
                if( hypo->nextDataIndex == numWords )
                {
                        StringSL test( hypo->wordHistory );

                        StringSL *found = uniqueHypo.find( &test );

                        if( found == 0 )
                        {
                                uniqueHypo.add( new StringSL( hypo->wordHistory
=> ) );

                                topList.push( hypo );
                        }
                }
                else
                {
                        hypo->propagate( &hypoPQ );

                        if( hypoPQ.count() > 1000 )
                                hypoPQ.removeEntriesfor( 0, 500 );
                }
        }

        // now fill ChoiceList
   int cnt=topList.count(); // maximum 40, can be less

   // Get memory for ChoiceList (of type AlphaHypAC:AlphaHypothesis)
```

```
        if( cnt )
        {
                // Make sure choice list is larger enough
                int i = choiceList->makeRoom( cnt );

                // process the data from topList[] class into a AlphaHypothesis
=> getting
                // SO_WORD id's for the digit strings, and move it into a AlphaH
=> ypoAC
                // the class for the choice list.
                // After this, we have our ContRecog choice list.
                // Have a look at AlphaHypothesis::AlphaHypothesis
                while( cnt-- )
                {
                        new( (void* )( &( (*choiceList)[ i++ ] ) ) )
                        AlphaHypothesis( recog, topList[ cnt ]->value(), zr );
                }
        }

        int j = choiceList->makeRoom( 1 );

        new( (void* )( &( (*choiceList)[ j ] ) ) ) AlphaHypothesis( recog, zr );

    return numWords;
}


// allocate memory and zero it
void Hypothesis::initMatrix( )
{
        // allocate memory
        for( int i = 0; i < NUMITEMS; i++ )
        {
                permutationScore[ i ] = new int[ NUMITEMS * NUMITEMS ];
                insertionScore[ i ] = new int[ NUMITEMS * NUMITEMS ];
                deletionScore[ i ] = new int[ NUMITEMS * 1 ];
        }

        // initialize matrix with 0's
        for( i = 0; i < NUMITEMS; i++ )
                for( int j = 0; j < NUMITEMS; j++ )
                        for( int k = 0; k < NUMITEMS; k++ )
                        {
                                deletionScore[i][ j * 1 ] = 0;
                                insertionScore[i][j * NUMITEMS + k] = 0;
                                permutationScore[i][j * NUMITEMS + k] = 0;
                        }
}


// read data from file into matrix
void Hypothesis::fillMatrix( int type, int numEmitted )
{
        char digitStr[20];
        int l = 0;
        long score;

        int lineSize = NUMITEMS * 20;

        // initialize with values from file
        char Line[NUMITEMS * 20];
        char *pLine;

        // Read matrix
        for( int i = 0; i < NUMITEMS; i++ )
        {
```

```
                    HYPO.CPP  10-20-95  3:11p
                                                        Page 6 of 7
        // Read the Main letter and the row of letters
        fgets( Line, lineSize, matrixFile );

        fgets( Line, lineSize, matrixFile );

        for( int j = 0; j < NUMITEMS; j++ )
        {
                // get the lines with scores
                if( fgets( Line, lineSize, matrixFile ) == NULL )
                                return;

                pLine = Line;
                pLine+=4;                       // skip first column

                for( int k = 0; k < numEmitted; k++ )
                {
                        // find a number
                        l = 0;

                        // skip anything different from digits
                        while( !isdigit( *pLine ) ) pLine++;

                        // read digits
                        while( isdigit( *pLine ) )
                                digitStr[ l++ ] = *pLine++;

                        digitStr[l] = 0;
                        score = atol( digitStr );

                        switch ( type ) {
                         case DELETION:
                                deletionScore[i][j*numEmitted + k] = 3*s
=> core;
                                break;

                         case INSERTION:
                                insertionScore[i][j*numEmitted + k] = sc
=> ore;
                                break;

                         case PERMUTATION:
                                permutationScore[i][j*numEmitted + k] =
=> score;
                                break;
                        }
                }
        }
        // take last empty line
        fgets( pLine, lineSize, matrixFile );
}


// read data from a matrix file
void Hypothesis::readMatrixFile( char *matrixName )
{
        char Line[200];
        int lineSize = 200;
        int matrixType = 0;

        // a name should have been defined via an open file dialog box
        if( matrixName[0]  == 0)
                return;

        // open in read mode
        matrixFile = fopen( matrixName, "rt" );

        assert( matrixFile );

        initMatrix();

        // there are 3 types of matrices
        for(int m = 0; m < NUMTYPE; m++)
        {
                fgets( Line, lineSize, matrixFile );

                if( !strncmp( Line, "Deletion", 8 ) )
                        matrixType = DELETION;
                else if( !strncmp( Line, "Permutation", 11 ) )
                        matrixType = PERMUTATION;
                else if( !strncmp( Line, "Insertion", 9 ) )
                        matrixType = INSERTION;

                // get empty line
                fgets( Line, lineSize, matrixFile );

                switch ( matrixType ){

                 case DELETION:
                        fillMatrix( DELETION, 1 );
                        break;

                 case PERMUTATION:
                        fillMatrix( PERMUTATION, NUMITEMS );
                        break;

                 case INSERTION:
                        fillMatrix( INSERTION, NUMITEMS );
                        break;
                }
        }

        fclose( matrixFile );
}

// smooth data
void Hypothesis::smoothLogMatrix( char *type, int numEmitted )
{
        int log1 = 0;
        int mirrorLog = 0;

        int** lettersLog;

        if( !strcmp( type, "Permutation" ) )
                lettersLog = permutationScore;

        else if ( !strcmp( type, "Insertion" ) )
                lettersLog = insertionScore;

        else return;

        // i == 1 and k == 1: ignore the silence case
        for( int i = 0; i < NUMITEMS; i++)
        {
                for( int j = 0; j < NUMITEMS; j++ )

                        for( int k = 0; k < numEmitted; k++)
                        {
                                // these values are all zero
                                if( i == k ) continue;
```

```
                                        log1  = letterslog[i][j * NUMITEMS + k ];

                                        mirrorlog = lettersLog[k][j *NUMITEMS + i];

                                        // log1 has no data or never occurred
                                        // mirrorlog has better data
                                        if(    log1 >= 92 && mirrorLog < 92
                                         && ( log1 - mirrorLog) > 15  )
                                                letterslog[i][j * NUMITEMS + k ] = mirro
=> rLog + 10;

                                        // both values are smaller than 92 and are prett
=> y close
                                        else if( log1 < 92 && mirrorlog < 92 )
                                                if( log1 > mirrorLog && ( log1 - mirrorl
=> og ) < 15  )
                                                        lettersLog[i][j * NUMITEMS + k ]
=>  = ( log1 + mirrorLog ) / 2;

                                }
        }

        int totallog = 0;

        int count = 0;

        char warning[100];

        for( i = 0; i < NUMITEMS; i++)
        {
                for( int k = 0; k < numEmitted; k++)
                {
                        // these values are all zero
                        if( i == k )continue;

                        totallog  = 0;

                        for( int j = 0; j < NUMITEMS; j++ )
                        {
                                if( letterslog[ i ][ j * NUMITEMS + k ] < 92 )
                                {
                                        totallog += lettersLog[ i ][ j * NUMITEM
=> S + k );

                                        count++;
                                }
                        }

                        if( count )     totallog = totallog / count;

                        count = 0;

                        // smooth the column
                        for( j = 0; j < NUMITEMS; j++ )
                        {
                                // if it is really off the average, assign the a
=> verage plus
                                // an offset
                                if( lettersLog[ i ][ j * NUMITEMS + k ] > totalL
=> og*8/5 )
                                        lettersLog[ i ][ j * NUMITEMS + k ] = to
=> talLog + 15;
                        }

                        if( lettersLog[i][ 0 + k ] > totallog )
                                lettersLog[i][ 0 + k ] = totallog;
                }
```

HYPO.CPP  10-20-95  3:11p

```
                }
}


// read data from a matrix file
void Hypothesis::readSpecialFile( char *specialName )
{
        char line[200];
        int lineSize = 200;

        char pattern[10];
        char emission[10];

        // a name should have been defined via an open file dialog box
        if( specialName[0]  == 0)
                return;

        // open in read mode
        specialFile = fopen( specialName, "rt" );

        assert( specialFile );

        while( fgets( line, lineSize, matrixFile ) != NULL )
        {
                sscanf( line, "%s %s", pattern, emission );

                StringSL test( pattern, emission );

                StringSL *found = specialHypo.find( &test );

                if( found == 0 )
                        specialHypo.add( new StringSL( pattern, emission ) );
        }

        fclose( specialFile );
}


// independent function
int compareChoice( const void* given, const void* test )
{
    return ((Choice* )test)->score - ((Choice* )given)->score;
}
```

Page 7 of 7

```
// HASHALFA.H
// Header file for spelling application classes
//
//                                                        CONFIDENTIAL

#ifndef hashalpha_h
#define hashalpha_h

#include "ordcoll.h"
#include "arrcoll.h"
#include "oneshot.h"
#include "stack.h"
#include "dragcpp.h"


// number of items in confusability matrix: silence, letters and digits
// and , . -
#define NUMITEMS    40
#define NUMTYPE          3

#define DELETION                         0
#define PERMUTATION                      1
#define BLOCKPERMUTATION         1
#define INSERTION                        2

class StringSL : public SLink< StringSL >
{
 public:
        char s[50];
        char emission[10];

        StringSL( char *ss ) { strcpy( s, ss ); }
        StringSL( char *ss, char *so ) { strcpy( s, ss ); strcpy( emission, so )
  => ; }
};

class StringHS : public HS< StringSL >
{
 public:
    StringHS( unsigned maxSize = 1024, unsigned initialSize = 1024 )
        : HS< StringSL >( maxSize, initialSize ) { }

    int compare( const SLinkBase*, const SLinkBase* ) const ;

 protected:

        unsigned keyHash( const SLinkBase* ) const ;
};


class AlphaHypothesis
{
 public:
        char phrase[ UTT_PROMPT_LENGTH ];
        char spelling[ UTT_PROMPT_LENGTH ];

        bool isNewWord;
        SD_WORD wordId;

        //    CityHypothesis* ch;
        AlphaHypothesis( AlphaHypothesis& );
        AlphaHypothesis( Recognizer*, AlphaHypothesis& );
        AlphaHypothesis( Recognizer*, char*, AlphaHypothesis& );
        AlphaHypothesis() { };

        int initfromPhrase();
                           HASHALFA.H  11-6-95  2:30p
```

```
        int initfromSpelling();
};

class AlphaHypoAC : public AC< AlphaHypothesis >
{
 public:
        AlphaHypoAC( unsigned initialsize = 5, unsigned incrementsize = 5 )
        : AC< AlphaHypothesis >( initialsize, incrementsize ) { }

        void clearUpNewWords( Recognizer* );
};

////////////////////////////////////////////////////////////////////////////
// General hypotheses propagation module
  =>                        //
////////////////////////////////////////////////////////////////////////////
#define MAX_WORD_HISTORY         16

class Hypothesis
{

 protected:
        int l2i( int l );
        int i2l( int l );

        void initMatrix();
        void fillMatrix( int type, int numEmitted );

 public:
        static OneShotHeap heap;
        static int* permutationScore[ NUMITEMS ];
        static int* insertionScore[ NUMITEMS ];
        static int* deletionScore[ NUMITEMS ];

        static char* data;
        static int dataLength;
        int nextDataIndex;

        char wordHistory[ MAX_WORD_HISTORY ];
        int nextWordIndex;

        int prevOperation;
        int curScore;

        Hypothesis();

        Hypothesis( Hypothesis& h );

        void propagate( PriorityQue<Hypothesis>* hpq );

        int& counts(int type, char mainL, char contextL, char emittedL);

        static int build( Recognizer*, AlphaHypoAC*, AlphaHypothesis&, int, int
  => );

        char* value() { return( wordHistory ); }

        void readMatrixFile( char *matrixName );
        void readSpecialFile( char *specialName );

        void smoothLogMatrix( char *type, int numEmitted );

};

                             Page 1 of 2
```

```
class Choice
(
 public:
        static OneShotHeap heap;

        char word[200];
        int score;

        Choice()
        (
                score = 0;
        );

);


class HypoPQ : public PriorityQue < Hypothesis >
(
 public:
        int checkStack();
        HypoPQ( CompareOCEntries cmpFunc, int initialSize = 40, int incrementSiz
  => e = 40 )
        : PriorityQue< Hypothesis >( cmpFunc, initialSize, incrementSize ) ( )
);

int compareChoice( const void* given, const void* test );


#endif
```

```
/* Module to load word list into an ordered collection
*/

#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "seq.h"
#include "ordcoll.h"
#include "arrcoll.h"
#include "oneshot.h"
#include "trie.h"

#define LETTERS         26

WordOC* WordOC::twoLetters[ LETTERS * LETTERS + 1 ];


// compare word
int wordCompare( const void* given, const void* test )
{
        return stricmp( (char* )given, (char* )test );
}

int boolWordCompare( const void* given, const void* test )
{
        return strncmpi( (char* )given, (char* )test, strlen( (char* )given ) );
}


bool WordOC::doWordsExist( char* pat, int numLetters )
{
        // 2 letter  words are put in special index
        WordOC *woc;

        // let one letter hypotheses live
        if( numLetters == 1)
                return TRUE;

        if ( numLetters == 2 )
        {
                // is pattern a 2 letter word
                woc = twoLetters[ LETTERS  * LETTERS ];

                int position;

                int found = woc->find( pat, boolWordCompare, &position );

                if( found != -1 )
                        return 1;
        }

        // 2 or more letters, maybe not a word
        woc = twoLetters[ getWordOCFromPat( pat ) ];

        int position;

        int found = woc->find( pat, boolWordCompare, &position );

        if( found != -1 )
                return 1;

        if( position != 0 )
        {
                if( 0 == strncmpi( pat, (*woc)[ position - 1 ], numLetters ) )
                        TRIE.CPP  9-22-95  4:24p
                        return 1;
        }
        if( position != woc->count() )
        {
                if( 0 == strncmpi( pat, (*woc)[ position ], numLetters ) )
                        return 1;
        }
        return 0;
}

int WordOC::getMatchingWords( WordOC* outOc, char* pat )
{
        // 2 letter  words are put in special index
        WordOC *woc = NULL;
        WordOC *woc2letters = NULL;

        int numLetters = strlen( pat );

        if ( numLetters == 2 ) {
                woc2letters = twoLetters[ LETTERS  * LETTERS ];
                woc = twoLetters[ getWordOCFromPat( pat ) ];
        }
        else
                woc = twoLetters[ getWordOCFromPat( pat ) ];

        int position;

        int found = woc->find( pat, boolWordCompare, &position );
        // -1 means NOT found
        if( found != -1 )
        {
                int len = strlen( pat );

                while( found > 0 )
                {
                        if( 0 == strncmpi( pat, (*woc)[ --found ], len ) )
                                outOc->add( (*woc)[ found ] );
                }
                int cnt = woc->count();

                while( position < cnt )
                {
                        if( 0 == strncmpi( pat, (*woc)[ position ], len ) )
                                outOc->add( (*woc)[ position ] );
                        position++;
                }

                // return now if 2 letter words were not looked up
                if (!woc2letters)
                        return woc->count();
        }

        // if 2 letter words need to be looked up
        if( woc2letters ){

                int found = woc2letters->find( pat, boolWordCompare, &position )
  );

                // -1 means NOT found
                if( found != -1 )
                {
                        int len = strlen( pat );

                        while( found > 0 )
                        {
                                if( 0 == strncmpi( pat, (*woc2letters)[ --found
                                 Page 1 of 3
```

```
    => ], len ) )
                                        outOc->add( (*woc2letters)[ found ] );
                        }
                        int cnt = woc2letters->count();

                        while( position < cnt )
                        {
                                if( 0 == strncmpi( pat, (*woc2letters)[ position
    => ], len ) )
                                        outOc->add( (*woc2letters)[ position ] )
    => ;
                                position++;
                        }

                        return( outOc->count() );
                } // if (found == -1)
        }

        return 0;
}

// print words
void WordOC::printWordOC()
{
        int numberOfWords = count();
        for( int i = 0; i < numberOfWords; i++ ) {
                printf( "%s\n", (*this)[ i ] );
        }
}

void WordOC::init()
{
        if( twoLetters[ 0 ] == 0 )
        {
                for( int i = LETTERS * LETTERS + 1; i--; )
                {
                        twoLetters[ i ] = new( &heap ) WordOC();
                }
        }
}

// Given a spelling return index in twoLetter array
// spelling can also be a pattern
int WordOC::getWordOC( char* pSpelling )
{
        assert( pSpelling && *pSpelling );
        int index = 0;

        if( 0 == *(pSpelling + 1) || 0 == *(pSpelling + 2) )
        {
                index = LETTERS * LETTERS;
        }
        else
        {
                // index in wordOC based on first 2 letters
                int c = tolower( *pSpelling );
                if( c >= 'a' && c <= 'z' )
                {
                        index = LETTERS * ( c - 'a' );

                        c = tolower( *(pSpelling + 1) );
                        if( c >= 'a' && c <= 'z' )
                        {
                                index += ( c - 'a' );
                        }
                }
```

```
                }
                return index;
}


// Given a spelling return index in twoLetter array
// spelling can also be a pattern
int WordOC::getWordOCFromPat( char* pSpelling )
{
        assert( pSpelling && *pSpelling );
        int index = 0;

        int c = tolower( *pSpelling );

        // for now same index as 2 letter words
        if( c >= '0' && c <= '9' )
                index = LETTERS * LETTERS;

        else if( c >= 'a' && c <= 'z' )
        {
                index = LETTERS * ( c - 'a' );

                c = tolower( *(pSpelling + 1) );
                if( c >= 'a' && c <= 'z' )
                {
                        index += ( c - 'a' );
                }
        }
        return index;
}


// class WordOC member function to add words in the right place of
// twoletters
void WordOC::addWord( char *pSpelling )
{
        WordOC* woc = twoLetters[ getWordOC( pSpelling ) ];

        int position;

        int found = woc->find( pSpelling, wordCompare, &position );

        if( found == -1 )
        {
                char *s = new( &heap ) char[ strlen( pSpelling ) + 1 ];

                strcpy( s, pSpelling );

                strlwr( s );

                woc->add( s, position );
        }
}

// Check spelling
int WordOC::checkSpelling(char *spell)
{
        char *p;
        for( p = spell; *p; p++ )
                if( !isalpha(*p) )
                        return 0;

        return 1;
}
#if 0
void main(int argc, char* argv[])
```

```
{
        FILE *infile;
        FILE * outFile;

        // my own local temporary wordOC
        WordOC *tmpWoc = new WordOC();

        char line [100];
        char spelling(50];
        char key(3];
        int length;
        int index;

        if (argc < 3)
                printf("\n USAGE: <Word List> < output >");

        inFile = fopen( argv(1], "rt" );
        outFile =fopen( argv(2], "wt" );

        // initialize twoLetter words
        WordOC::init();

        // read in file
        while( fgets( line, 100, infile ) )
        {
                sscanf( line, "%s", spelling );
                int noSpecialWord = WordOC::checkSpelling( spelling );

                if (noSpecialWord) {
                        length = strlen( spelling );
                        if ( length < 2 )
                                strcpy( key ,spelling );
                        else
                        {
                                strncpy( key ,spelling , 2 );
                                key(2] = 0;
                        }
                        WordOC::addWord( spelling );
                }
        }

        // Given a word return list
        WordOC::getMatchingWords( tmpWoc, "no" );
        tmpWoc->printWordOC();

        //If( !WordOC::doWordsExist( "xy" ) )
        //      printf( "Nothing" );

        fclose(infile);
        fclose(outfile);
}
#endif
```

```
// hypo.h

#ifndef hypo_h
#define hypo_h

#include "dragcpp.h"
#include "stack.h"
#include "oneshot.h"


#define BAD     (1000)


class AlphaHypothesis
{
 public:
    bool isNewWord;
    SD_WORD word;
    long value;

    char spelling[ UIT_PROMPT_LENGTH ];
    char phrase[ UIT_PROMPT_LENGTH ];

    ZipHypothesis( Recognizer*, long zip, char* stateName );
    ZipHypothesis( Recognizer*, const char*, const char* );
};

class AlphaHypoAC : public AC< AlphaHypothesis >
{
 public:
    ZipHypoAC( unsigned initialsize = 5, unsigned incrementsize = 5 )
    : AC< ZipHypothesis >( initialsize, incrementsize ) { }

    void cleanUpNewWords( Recognizer* );
};


///////////////////////////////////////////////////////////////////////


class Hypothesis
{
 protected:
//    static int confusionScore[ 100 ];

 public:
    static OneShotHeap heap;

    int wordHistoryArray[ 5 ];

    long currentScore;
    long nextWordIndex;


    Hypothesis() { currentScore = BAD; nextWordIndex = 0; }

    Hypothesis( Hypothesis& h ) {
        currentScore = h.currentScore;
        nextWordIndex = h.nextWordIndex;
        memcpy( wordHistoryArray, h.wordHistoryArray, sizeof( wordHistoryArray )
  *>  );
    }

    void propagate( PriorityQue<Hypothesis>* hpq, int* data, int stateId );

    long value() {
```

HYPO.H  11-6-95  2:47p

```
            return( wordHistoryArray[ 0 ] * 10000L + wordHistoryArray[ 1 ] * 1000L
                    + wordHistoryArray[ 2 ] * 100L + wordHistoryArray[ 3 ] * 10L
                    + wordHistoryArray[ 4 ] );
    }
};


extern void buildHypothesis( Recognizer*, AlphaHypoAC*, char*, char*, int );


#endif // hypo_h
```

Page 1 of 1

```
/////////////////////////////////////////////////////////////////////////
//
//
/*

CHOICE.CPP

Main module: Windows loop and functionality of the application

A wide variety of functions can be found here:


FUNCTIONS
.........

1. Functions declared in a class and defined in WAPP.CPP:


return          Class                            member function
Value
......          .....                            ...............


unsigned        WordIdToCustomCityHS::keyHash( const SLinkBase* slb ) const
int             WordIdToCustomCityHS::compare( const SLinkBase* slb1, const SLin
  => kBase* slb2 ) const


FindFilePath               ::FindFilePath( char* fspec, char* p, int flag )
bool            FindFilePath            ::findFirstPrimitive()
bool            FindFilePath            ::findFirst()
bool            FindFilePath            ::findNext()


void            DlgWin                           ::onSpeech( HWND hwnd, SD_CHANNE
  => L ch )
void            DlgWin                           ::onSysCommand( HWND hwnd, UINT
  => cmd, int, int )
void            DlgWin                           ::onCommand( HWND hwnd, UINT cmd
  => , HWND, UINT )
void            DlgWin                           ::onTimerDrawVU( HWND hwnd, UINT
  =>  id )
void            DlgWin                           ::onDestroy( HWND hwnd )
void            DlgWin                           ::itsBad( HWND hwnd, UttChannel*
  =>  toMe )
void            DlgWin                           ::itsGoodAdapt( AlphaHypothesis&
  =>  ch )
void            DlgWin                           ::reportError( int code, char fa
  => r *message )

bool            DlgWin                           ::onInitDialog( HWND hDlg, HWND,
  =>  long )
bool            DlgWin                           ::isCancelBadLabelGoToSleep( HWN
  => D hwnd )
bool            DlgWin                           ::isCertain( SD_WORD testWord )

void SD_CALLBACK _export DlgWin ::postSpeechEvent( SD_CHANNEL ch, SD_CHANNEL_EVE
  => NT eventIn )
BOOL FAR PASCAL  _export DlgWin ::wndProc( HWND hwnd, UINT message, WPARAM wPara
  => m, LPARAM lParam )
```

CHOICE.CPP 10-26-95 3:18p

```
void            AppWin                           ::onCommand( HWND hWnd, UINT cmd
  => , HWND, UINT )
void            AppWin                           ::onDestroy( HWND )
void            AppWin                           ::onNCDestroy( HWND )
void            AppWin                           ::onClose( HWND hWnd )
bool            AppWin                           ::onCreate( HWND hwnd, CREATESTR
  => UCT FAR* )

long FAR PASCAL  _export AppWin  ::wndProc( HWND hwnd, UINT message, WPARAM wPara
  => m, LPARAM lParam )



      2. Main function

int PASCAL WinMain( HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR lpszCmdLine,
  =>  int nCmdShow )


   CLASSES declared  in WAPP.CPP
   .............................


      class DlgWin
      class FindFilePath
      class AppWin

      struct ConfigCommands

  =>                                                                          */
/////////////////////////////////////////////////////////////////////////


// INCLUDE


#include <windowsx.h>
#include <string.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <bios.h>
#include <io.h>
#include <stdio.h>
#include <fcntl.h>
#include "dragcpp.h"
#include "choice.h"
#include "slidectl.h"


                                                                          ex
  =>
  => tern "C" {
    #include "fepext.h"

  =>                                                                      }


#include <mmsystem.h>
#include <ddeml.h>
extern long computeMemoryUse();

// This is the instance of the DgnSlideControl
```

Page 1 of 17

```
DgnSlideControl         slideControl;
FARPROC lpfnTrain;


extern HINSTANCE        hInstance;      // when we call DgnSlideControl::libMain
  => Init()
HINSTANCE                       hInstance;      // hInstance must be initialized
  => ...


#include "hashalfa.h"
#include "trie.h"
#include "deamon.h"
#include "traincpp.h"


// DEFINE

#define WAKEUP_STATE                    0
#define ALPHA_STATE                             1
#define WORD_STATE                              2

#define FEP_STARTOFSPEECHTHRESH 13
#define FEP_ENDOFOSPEECHTHRESH  12

#define handleMessage( message, hWnd, wParam, lParam, fn ) \
 case (message) : returnValue = HANDLE_ ## message( (hWnd), (wParam), (lParam),
  => (fn) ); break;



// STATIC VARIABLES

static char femCityVoc[]        = "ctall.vof";
static char femalphaVoc[]       = "alpha.vof";
static char malCityVoc[]        = "system.voc";
static char malalphaVoc[]       = "alpha.vom";
static char wordVoc[]           = "system.voc";

static char origCityUser[ 128 ];
static char tmpCityUsr[] = "tmpcty.usr";

// global because used in train.cpp
char* cityUser = origCityUser;
char* cityVoc = 0;
char* alphaVoc = 0;

static char userPathString[ 256 ] = ".";

static char bluePrefix[ 6 ] = "00";
static char orangePrefix[ 6 ] = "09";

static char lastZipOutCommPort[ UTT_PROMPT_LENGTH ] = "";
static char lastStateOutCommPort[ UTT_PROMPT_LENGTH ] = "";
static char lastCityOutCommPort[ UTT_PROMPT_LENGTH ] = "";


BOOL isInList = FALSE;


FILE* errFile;

extern char stateAliasString[];

static char* sayPrompt[] = {
        "Say: WakeUp",
        "Say: Letters: a b ..",
        "Say a word",
};

static SD_STATE tmpState;

//////////////////////////////////////////////////////////////////////
//
//      CLASS DlgWin
//
//
//      Purpose: Drives the application.
//
//              The member function onSpeech deals with the recognition
//              of the zip code, state name and city name. It applies
//              correction mechanisms when the proposed answers of the three
//              components do not lock together.
//
//////////////////////////////////////////////////////////////////////



//////////////////////////////////////////////////////////////////////
//
//      DlgWin variables and classes defined in the DlgWin Class are
//      made global.
//
//////////////////////////////////////////////////////////////////////



HWND            DlgWin::hWnd;
bool            DlgWin::messageHandled;
SpeechTask*     DlgWin::speechTask;
Recognizer*     DlgWin::recog;                  // recognizer, wrapper around SD
  => API (dragcpp.*)
UttChannel*     DlgWin::channel;
bool            DlgWin::sawError;
bool            DlgWin::ignoreErrors;
char*           DlgWin::name = "Choice List App";

char            DlgWin::firstChoiceArray[20][ 16 ];
int             DlgWin::nChoice;
int             DlgWin::nPattern;

// storage of recognized patterns and their distance
Pattern         DlgWin::patternDistances[MAX_RECOG_CHOICES];


UttChannel      DlgWin::alphaUtt;
UttChannel      DlgWin::wordUtt;

AlphaHypoAC     DlgWin::alphaHypoAC( 64 );


AlphaHypothesis DlgWin::alphaResult;
char                    DlgWin::alphaSpelling[ UTT_PROMPT_LENGTH ];
char                    alphaSpellingCsr[ UTT_PROMPT_LENGTH ];

bool            DlgWin::wasListening;
bool            DlgWin::fixError = 1;
long            DlgWin::numErrors;
short           DlgWin::wordPause;
short           DlgWin::startThresh;
short           DlgWin::endThresh;


int                     DlgWin::appState;                  // zips, city, s
  => tate, error
SD_WORD         DlgWin::cancelWord = 0;
SD_WORD         DlgWin::badLabelWord = 0;
SD_WORD         DlgWin::goToSleepWord = 0;
SD_WORD         DlgWin::wakeUpWord = 0;
SD_WORD         DlgWin::someZipWord = 0;
SD_STATE        DlgWin::testState;


char            DlgWin::buf[ UTT_PROMPT_LENGTH ];       // scratch buf

//////////////////////////////////////////////////////////////////////
//
// CLASS CustomCitySL
//
// Purpose:
//
//      Process Command Words and Provinces
//
//////////////////////////////////////////////////////////////////////

static char* setUserPath( char* value )
{
        for( char *s = value; *s && !isspace( *s );     ++s );

        *s = '\0';

        strcpy( userPathString, value );

        return 0;
}


// Helper function

static char* setFepPar( WORD id, char* value )
{
        for( char *s = value; *s && !isspace( *s );     ++s );

        *s = '\0';

        if( *value )
        {
                WORD v;

                v = (WORD )atol( value );

                Fep_SetPar( SUPER_USER, id, v );

                return 0;
        }
        return "Par Not Set";
}


// Helper function
```

```
static char* setStartOfSpeech( char* value )
{
        if( setFepPar( FEP_STARTOFSPEECHTHRESH, value ) )
                return "StartOfSpeech Not Set";

        return 0;
}


// Helper function

static char* setEndOfSpeech( char* value )
{
        if( setFepPar( FEP_ENDOFOSPEECHTHRESH, value ) )
                return "EndOfSpeech Not Set";

        return 0;
}


// Helper function

static char* setNothing( char* )
{
        return 0;
}


//////////////////////////////////////////////////////////////////////
//
// class FindFilePath
//
// Purpose:
//
//              Find file using PATH variable
//
//////////////////////////////////////////////////////////////////////


class FindFilePath
{
 protected:

        static char defaultPath[];
        static char defaultFileSpec[];

        char *pathString, *path;
        char *fileSpec;
        int attribFlag;

        struct ffblk attrib;

        bool findFirstPrimitive();

 public:

        FindFilePath( char* fspec, char* p =0, int flag =FA_NORMAL );

        ~FindFilePath()
        {
                if( pathString != defaultPath )
                        delete pathString;

                if( fileSpec != defaultFileSpec )
```

```
                        delete fileSpec;
        }

        bool FindFilePath::findFirst();
        bool FindFilePath::findNext();

        char* filename() { return attrib.ff_name; }
};


char FindFilePath::defaultFileSpec[] = "*.*";
char FindFilePath::defaultPath[] = ".\0";


// Constructor

FindFilePath::FindFilePath( char* fspec, char* p, int flag )
{
        attribFlag = flag;

        if( p )
        {
                path = pathString = new char[ strlen( p ) + 2 ];
                strcpy( pathString, p );
                for( p = pathString; *p; ++p )
                        if( *p == ';' )
                                *p = '\0';

                *++p = '\0';
        }
        else
                pathString = path = defaultPath;

        if( fspec )
        {
                fileSpec = new char[ strlen( fspec ) + 1 ];
                strcpy( fileSpec, fspec );
        }
        else
                fspec = defaultFileSpec;
}

// Helper function

bool FindFilePath::findFirstPrimitive()
{
        for(;;)
        {
                if( *path == '\0' )
                        return FALSE;

                char* fSpec = new char[ strlen( path ) + strlen( fileSpec ) + 2
   => ];

                char* p = stpcpy( fSpec, path );

                if( p>fSpec && *(p-1) != ':' && *(p-1) != '\\' && *(p-1) != '/'
   => )
                        *p++ = '/';

                p = stpcpy( p, fileSpec );

                path +=  + strlen( path ) + 1;

                if( !findfirst( fSpec, &attrib, attribFlag ) )
                {
```

```
                        delete fSpec;
                        return TRUE;
                }
                delete fSpec;
        }
}

// Helper function

bool FindFilePath::findFirst()
{
        path = pathString;

        return findFirstPrimitive();
}


// Helper function

bool FindFilePath::findNext()
{
        if( !findnext( &attrib ) )
                return TRUE;

        return findFirstPrimitive();
}


// initialisation function for DlgWin class (NOT constructor)
// the global variable appState is set to ALPHA_STATE


bool DlgWin::onInitDialog( HWND hDlg, HWND, long )
{
        hWnd = hDlg;

        speechTask = new SpeechTask( name, reportError );


        //              Memory debug stuff

        //      int16 track;
        //
        //      speechTask->getPar( "track-mem", &track );
        //
        //      speechTask->setPar( "track-mem", (short )!track );

        recog = 0;


        //      Fep_Setpar( SUPER_USER, 13, 87 );
        //      Fep_Setpar( SUPER_USER, 12, 58 );

        //        took out parseString

        FindFilePath ffPath( "*.un?", userPathString );

        // set up the usr listbox
        if( ffPath.findFirst() )
        {
                do
                {
                        SendDlgItemMessage( hDlg, USR_COMBO_ID, CB_ADDSTRING, 0,
   =>  (DWORD) ffPath.filename() );
                }
                while( ffPath.findNext() );
```

```
	}

	appState = WORD_STATE;

	int16 comptln;
	//	speechTask->getPar( "computation", &comptln );
	speechTask->setPar( "computation", (short )15 );
	//	speechTask->setPar( "adapt-tolerance", (short )3 );

	SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLRANGE, 0, 0x0
=> 1000060L );
	SendDlgItemMessage( hDlg, WORD_PAUSE_SLIDE_ID, WM_SETSCROLLPOS, 0x00a0,
=> 1 );

	SendDlgItemMessage( hDlg, START_THRESH_SLIDE_ID, WM_SETSCROLLRANGE, 0, 0
=> x00800020L );
	SendDlgItemMessage( hDlg, START_THRESH_SLIDE_ID, WM_SETSCROLLPOS, 0x0050
=> , 1 );

	SendDlgItemMessage( hDlg, END_THRESH_SLIDE_ID, WM_SETSCROLLRANGE, 0, 0x0
=> 0700010L );
	SendDlgItemMessage( hDlg, END_THRESH_SLIDE_ID, WM_SETSCROLLPOS, 0x0030,
=> 1 );

	// initialize twoLetter words in ordered array to pointers of char
	WordOC::Init();

	// read in file with words
	char line[100];		// line of file
	char buffer[100];	// for error message

	char spelling[50];

	char wordFileName[] = "words.lst";

	// global
	errFile = fopen( "letters.log", "a" );

	FILE* infile = fopen( wordFileName, "r" );
	// Error checking
	if (infile == NULL) {
		sprintf( buffer, "Unable to open file %s\n",
				wordFileName );

		MessageBox( hWnd, buffer, "CHOICE-app Error",
					MB_TASKMODAL | MB_ICONSTOP | MB_OK );
		EndDialog( hWnd, 0 );

		return FALSE;
	}

	// Read in the list file
	while( fgets( line, 100, infile ) )
	{
		sscanf( line, "%s", spelling );

		// do not deal with words with spaces etc for now
		int noSpecialWord = WordOC::checkSpelling( strlwr( spelling ) );

		if (noSpecialWord)
			// Load words in WordOC
			WordOC::addWord( spelling );

	}
	fclose( infile );
```

CHOICE.CPP 10-26-95 3:18p

```
		// load matrix into statics
	Hypothesis *hypo = 0;
	hypo = new( &Hypothesis::heap ) Hypothesis;

		// hard coded matrix name
		hypo->readMatrixFile( "matrix.log" );

		// smooth logs
		hypo->smoothLogMatrix( "Permutation", NUMITEMS );
		hypo->smoothLogMatrix( "Insertion", NUMITEMS );

			//read in special cases
		hypo->readSpecialFile( "special.txt" );


		return TRUE;
}


// Dragon speech driver error handler.
// Immediately after having

void DlgWin::reportError( int code, char far *message )
{
		sawError = YES;
		if( ignoreErrors )
			return;		// prevent recursion
		ignoreErrors = YES;

		// Throw up an error message
		char buffer[ 512 ];
		sprintf( buffer, "An error has occurred\n  code = %d\n  message = %s",
						code, message );

		//	if( stricmp( message, "Bad token" ) && strncmp( message, "Invalid
  => word handle:", 18 ) )
		//	{
		MessageBox( hWnd, buffer, "Choice SDAPI Error",
							MB_TASKMODAL | MB_ICONST
  => OP | MB_OK );

		// terminate...  kill the hTopParentWindow
		//if( hWnd )
		//	PostMessage( hWnd, WM_CLOSE, 0, 0 );
		//	}
		ignoreErrors = 0;
}


/////////////////////////////////////////////////////////////////////
/* postSpeechEvent()

Callback function, handed to the microphone channel when it was opened.

We do not process any data in this function. We want to get out of this
interrupt as soon as possible.

Its place in the control flow:

Speech --> Board digitizes --> Multimedia (MM) layer discovers speech, and
interrupts other Windows activities --> dragdev, the dll that deals
with the front end (FEP), gets data from MM and decides whether we have
speech or not(Utterance detection).
If so, dragdev stores the utterance in its queue and it also calls
postSpeechEvent. This function was handed to the channel when we opened
```

Page 5 of 17

```
the channel. In this way the microphone knows to use this callback function
to get into the application for recognition.

After we leave the interrupt we send ourselves a PostMessage UM_ChannelStart.
When UM_ChannelStart is processed we direct the application to the function
onSpeech().

Make sure there is a start of utterance message for any utterance.
It was observed that having utterance and the related utterance start
message were out of sync. If PeekMessage return false, we will
make sure to put a UM_CHANNEL_START message in. This makes it more robust
to keep he number of utterances and CHANNEL_STARTS in sync.

A similar trick is applied in onSpeech() at EndOfSwitch.
*/
//////////////////////////////////////////////////////////////////////////


void SD_CALLBACK _export DlgWin::postSpeechEvent( SD_CHANNEL ch, SD_CHANNEL_EVEN
  => T eventIn )
{
        assert( eventIn == SD_CHANNEL_START );

        MSG msg;

        if( 0 == PeekMessage( &msg, hWnd, UM_CHANNELSTART, UM_CHANNELSTART, PM_N
  => OREMOVE | PM_NOYIELD ) )
                //??
                PostMessage( hWnd, UM_CHANNELSTART, (WORD )ch, 0 );
} // EventHandler


#define HANDLE_UM_CHANNELSTART(hwnd, wParam, lParam, fn) \
((fn)(hwnd, (SD_CHANNEL )wParam), 1L)


#define HANDLE_UM_COMM_CHECK_MSG( hwnd, wParam, lParam, fn ) \
((fn)(hwnd, (Comm* )lParam), 1L)


#define HANDLE_UM_COMM_SEND_SYNC_MSG( hwnd, wParam, lParam, fn ) \
((fn)(hwnd, (Comm* )lParam), 1L)



static long right=0, wrong=0, predicted=0, addressConfidence=100;


extern const char* ltoZip( long num );

// Helper function

void DlgWin::itsBad( HWND hwnd, UttChannel* toMe )
{
        // Stijn suggests: bad counter can be set and brought to screen here.
        // the zip state recognition failed.
        recog->beep( "bad.wav" );

        SetWindowText( GetDlgItem( hwnd, ALPHA_CHOICE_LIST ), "no recognition" )
  => ;
        SetWindowText( GetDlgItem( hwnd, CHOICE_LIST_ID ), "" );
```

```
        recog->giveAwayUtt( toMe );

        toMe->setPrompt( "" );

        appState = WORD_STATE;
}


void DlgWin::itsGoodAdapt( AlphaHypothesis& ah )
{
        //    assert( zh->hasIntegrity() );

        // Do something here with good answer, maybe output via DDE client
        // ...

        // lets adapt...
        recog->setVoc( alphaVoc );
        recog->setChannel( &alphaUtt );
        recog->setPrompt( ah.phrase );
        recog->contAdapt();

        recog->setChannel( channel );

        SetWindowText( GetDlgItem( hWnd, CHOICE_LIST_ID ), ah.spelling );

        static int count = 0;
        if( count++ == 4 )
        {
                count = 0;
                static writeCount = 0;
                static long write[ 4 ];

                char num[ 20 ];
                ltoa( write[ writeCount ]=computeMemoryUse(), num, 10 );
                SetWindowText( GetDlgItem( hWnd, MEMORY_TEXT_ID ), num );

                if( ++writeCount == 4 )
                {
                        writeCount = 0;
                        FILE * fout = fopen( "memuse.log", "a+" );
                        if( fout )
                        {
                                fprintf( fout, "%ld\n", write[ 0 ] );
                                fprintf( fout, "%ld\n", write[ 1 ] );
                                fprintf( fout, "%ld\n", write[ 2 ] );
                                fprintf( fout, "%ld\n", write[ 3 ] );
                                fclose( fout );
                        }
                }
        }
}


OneShotHeap Choice::heap( 4096 );

//////////////////////////////////////////////////////////////////////////
//
//      onSpeech is the heart of the solution.
//
//////////////////////////////////////////////////////////////////////////

void DlgWin::onSpeech( HWND hwnd, SD_CHANNEL ch )
{
        // get the utterance handle; do not wait for the end of the
        // utterance since the recognition can proceed in parallel
        // with the utterance collection


        static bool found = FALSE;

        assert( recog );

        // notify the dragcpp classes for speech event
        if( recog->notifyChannel( ch, SD_CHANNEL_START ) )
        (
                // appState is initialized to WORD_STATE
                switch( appState )
                (
                 case WORD_STATE:
                        (
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_RES
=> ETCONTENT, 0, 0);
                                SendDlgItemMessage( hwnd, ALPHA_CHOICE_LIST, LB_
=> RESETCONTENT, 0, 0);
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_RES
=> ETCONTENT, 0, 0);
                                SendDlgItemMessage( hwnd, TEMP_STATE_LIST_ID, LB
=> _RESETCONTENT, 0, 0);

                                // Get word vocabulary
                                recog->setVoc( cityVoc );

                                recog->setState( "Dictation");

                                if( 0 == recog->recog() )
                                ( \
                                        // get top choice
                                        for (int i = 0; i < recog->resultCount()
=> ; i++)
                                        (
                                                recog->resultName( buf, sizeof(
=> buf ), i  );

                                                // store first 2 choice list cho
=> ices
                                                if( i < 2)
                                                (
                                                        nChoice = i;
                                                        // keep patterns from ch
=> oice list around to be
                                                        // inserted later
                                                        strcpy( firstChoiceArray
=> (i), buf );
                                                )

                                                // display top choice on screen
                                                SendDlgItemMessage( hwnd, CHOICE
=> _LIST_ID, LB_ADDSTRING, 0, (DWORD) buf);
                                        )

                                        recog->giveAwayUtt( &wordUtt );

                                        appState = ALPHA_STATE;
                                )
                                else (
                                        recog->killUtt();

                                        appState = WORD_STATE;
                                )
```

```
                        ) // WORD_STATE
                        break;


                case ALPHA_STATE:
                        (
                                // reset variables to NULL and
                                // remove hypotheses from previous looping

                                alphaResult.spelling[ 0 ] = '\0';
                                alphaResult.phrase[ 0 ] = '\0';
                                alphaSpelling[ 0 ] = '\0';

                                int wasGoodZipPhrase = 0;
                                int numWords = 0;

                                alphaHypoAC.removeAll();

                                // Get alpha vocabulary
                                recog->setVoc( alphaVoc );
                                // Get digitOcolor state
                                recog->setState( "start" );

                                SendDlgItemMessage( hwnd, ALPHA_CHOICE_LIST, LB_
=> RESETCONTENT, 0, 0);
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_RES
=> ETCONTENT, 0, 0);
                                SendDlgItemMessage( hwnd, TEMP_STATE_LIST_ID, LB
=> _RESETCONTENT, 0, 0);

                                // start recognition
                                if( 0 == recog->contRecog() )
                                (
                                        recog->resultName( alphaResult.phrase, s
=> izeof( alphaResult.phrase ) );

                                        // returns length of phrase
                                        wasGoodZipPhrase = alphaResult.initFromP
=> hrase();

                                        strncpy( alphaSpelling, alphaResult.spel
=> ling, wasGoodZipPhrase );
                                        alphaSpelling[ wasGoodZipPhrase ] = '\0'
=> ;

                                        strcpy(alphaSpellingCsr, alphaSpelling )
=> ;
                                        strcat(alphaSpellingCsr, " CSR" );
                                        SendDlgItemMessage( hwnd, TEMP_STATE_LIS
=> T_ID, LB_ADDSTRING, 0, (DWORD) alphaSpellingCsr );
                                )

                                // Set state to zipcommandwords
                                recog->setState( "zipcommandwords" );

                                if( wasGoodZipPhrase )
                                        wasGoodZipPhrase = Hypothesis::build( re
=> cog, &alphaHypoAC, alphaResult, wasGoodZipPhrase, 150 );


                                //-----------------------------------------------
=> -----------
                                // RE-RECOGNIZE against phrases.
                                // Recognition in Alpha Voc in zipcommandwords
                                //-----------------------------------------------
```

CHOICE.CPP 10-26-95 3:18p

Page 8 of 17

```
=> ............

=> cognized word

=>  WORD state

=> ] ) <= 2 )
=> Array[ l ] );

=> cArray( l ], 3 );

=>  stored in buf
=> c, buf );

=> ern
=> t();

=> s; j++ )

=> ;
=> tmpWoc)[ j ] );

=> g->wordId( wordSpelling )) == 0 );

=> adds the WORD_ID to the state
```

```
        if( recog->recog() == 0 )(

                SD_WORD wordId = 0;

                // temporary wordOC
                static WordOC *tmpWoc = new WordOC();

                // Get word vocabulary
                recog->setVoc( cityVoc );

                // Create new state
                tmpState = recog->newState( "tmpWord" );

                // Assert new Word state
                assert( tmpState );

                // Set new Word state
                recog->setState( tmpState );

                // get 3 letter pattern of originally re

                // for robustness

                // nChoice set after first word recog in

                for( int l = 0; l <= nChoice; l++ )
                (
                        if( strlen( firstChoiceArray[ l

                                strcpy( buf, firstChoice

                        else
                        (
                                strncpy( buf, firstChoic

                                buf[3] = 0;
                        )
                        // Get list of words for pattern

                        WordOC::getMatchingWords(  tmpWo

                        // number of word with this patt

                        int numberOfWords = tmpWoc->coun

                        // Add words to temp state
                        for( int j = 0; j < numberOfWord

                        (
                                char wordSpelling[ 100 ]

                                strcpy( wordSpelling, (*

                                int isNewWord =
                                        ( (wordId = reco

                                // buildWord builds and
```

```
//
  => ing,"**");
//
  => ge( hwnd, TEMP_STATE_LIST_ID, LB_ADDSTRING, 0, (DWORD) wordSpelling);

  => to the state

  => wordId );

  => ndDlgItemMessage( hwnd,

  =>                            TEMP_STATE_LIST_ID,

  =>                            LB_FINDSTRING,

  =>                            0, (DWORD) wordSpelling) )

  => ge( hwnd,

  =>                            TEMP_STATE_LIST_ID,

  =>                            LB_ADDSTRING,

  =>                            0, (DWORD) wordSpelling);

#if 0
  =>  WORD STATE to state

#endif

  => dd words in from

  =>  output
  => buf ), i );

  => onces
  => ern, buf );

  => tance( i );
```

```
                                if( isNewWord ) (
                                        strcat(wordSpell

                                        SendDlgItemMessa
                                        continue;
                                )
                                else (
                                        // add the word

                                        recog->addWord(

                                        if( LB_ERR == Se

                                        SendDlgItemMessa

                                        numWords++;
                                )
                                assert( wordId != 0 );

                        ) // for NumberOfWords
                )
                // Add first choice of discrete recog in

                wordId = recog->wordId( firstBuf );

                recog->addWord( wordId );

                // inspect choice list of patterns and a

                // words.lst

                nPattern = recog->resultCount();

                int penalty = 3;

                for( int i = 0; i < nPattern; ++i )
                (
                        // fill buffer with recognizer's

                        recog->resultName( buf, sizeof(

                        // store patterns and their dist

                        strcpy( patternDistances[i].patt

                        int distance  = recog->resultDis
```

```
=> n the choices
=> hoice list the worse



=> ance = distance + penalty;


=> ance = distance;

=>  strings on screen
=> CHOICE_LIST, LB_ADDSTRING, 0, (DWORD) buf);

=>  stored in buf
=> c, buf );

=> ern
=> t();


=> s; j++ )

=> ;

=> tmpVoc)[ j ] );

=> d=recog->wordId( wordSpelling )) == 0 );

=> adds the WORD_ID to the state

//
=> ing,"***");
//
=> ge( hwnd, TEMP_STATE_LIST_ID, LB_ADDSTRING, 0, (DWORD) wordSpelling);


=> to the state
=> wordId );

=> ndDigItemMessage( hwnd,
                        TEMP_STATE_LIST_ID,
=>
                        LB FINDSTRING,
=>
```

```
// sometimes the distance betwee
// is zero. The further on the c

if( i > 0 && distance == 0 )
{
        penalty += 3;
        patternDistances[i].dist
}
else
        patternDistances[i].dist

// display choice list of letter
SendDigItemMessage( hwnd, ALPHA_

// Get list of words for pattern
WordDC::getMatchingWords(  tmpVo

// number of word with this patt
int numberOfWords = tmpVoc->coun

// Add words to temp state
for( int j = 0; j < numberOfWord
{
        char wordSpelling[ 100 ]

        strcpy( wordSpelling, (*

        int isNewWord = ( (wordI

        // buildWord builds and
        if( isNewWord ) {
                strcat(wordSpell

                SendDigItemMessa

                continue;
        }
        else {
                // add the word

                recog->addWord(

                if( LB_ERR == Se
```

CHOICE.CPP  10-26-95  3:18p

```
=>                    0, (DWORD) wordSpelling) )

=> ge( hwnd,
                      TEMP_STATE_LIST_ID,
=>
                      LB_ADDSTRING,
=>
                      0, (DWORD) wordSpelling);
=>







=> OF_WORDS_ID), numW );

=>  state


=> putation", (short )30 );

=> cognizer
=> nnel();



=> ded score of pattern
=> Q( compareChoice );



=> rdUtt with temp state



=> ltCount();
=> ist and put on  screen
```

```
                                SendDigItemMessa



                                numWords++;
                        }
                        assert( wordId != 0 );
                } // for NumberOfWords

                tmpVoc->removeAll();
        } // for

        static char numW[20];
        itoa( numWords, numW, 10 );
        SetWindowText( GetDigItem( hwnd, NUMBER_

        // make sure there are words in the temp
        if ( numWords )
        {
                if( numWords > 500 )
                        speechTask->setPar( "com

                // put wordUtt in channel for re
                UttChannel *chan = recog->getCha

                // put wordUtt in channel
                recog->setChannel( &wordUtt );

                // order best choice based on ad
                static PriorityQue< Choice > choiceP

                // remove previous candidates
                choicePQ.removeAll();

                Choice::heap.empty();

                // do discrete recognition on wo
                // Voc is set to cityVoc
                if ( 0 == recog->recog() )
                {
                        int number = recog->resu
                        // loop through choice l
                        for( int i = 0; i < numb
```

Page 9 of 17

CHOICE.CPP 10-26-95 3:18p

Page 10 of 17

```
=> er; ++i )                                                 (

=> ords to the screen
=> e( buf, sizeof( buf ), i );

=> ttern and add distance
=> n < nPattern; n++ )

=> f,
=>              patternDistances[n].pattern,
=>              strlen( patternDistances[n].pattern ))         )

=> recog->resultDistance( n );

=>   new( &heap ) Choice();

=> word, buf );

=> distance +
=>                      patternDistances[n].distance;

=> hoice );

=> nPattern loop and get next
=> tem


=> )



=> recog->resultDistance( n );

=>   new( &heap ) Choice();

=> word, buf );

=> y
=> distance + 150;

=> hoice );
```

```
// print those w
recog->resultNam

// search for pa
for( int n = 0;
{
if( !strncmp( bu


{
int distance =

Choice *choice =

strcpy( choice->

choice->score =


choicePQ.push( c

// break out of
// choice list i

found = TRUE;

break;
} // if( strcmp

} // nPattern

if( !found )
{
int distance =

Choice *choice =

strcpy( choice->

// 150 is penalt
choice->score =

choicePQ.push( c
```

```
=> creen
=>  bottom of stack


=> , CHOICE_LIST_ID,
=>                    LB_ADDSTRING,
=>                    0,
=>                    (DWORD ) choicePQ[ number ]->word);

=> oice list


=> f not 10 words

=> mber; k > 0; k-- )

=> ordOC *tmpWoc = new WordOC();

=> ist of words for pattern stored in buf
=> getMatchingWords(  tmpWoc,
=>

=> r of word with this pattern
=> erOfWords = tmpWoc->count();

=> ords to choice list
=>  j = 0; j < numberOfWords; j++ )

=> er < 9 )

=> dSpelling( 100 );

=> wordSpelling, (*tmpWoc)[ j ] );
```

```
                )
                found = FALSE;
        } // for number

        i = number-1;
        // Put final choice on s
        // Best words are at the
        while( number-- )
        {
        SendDlgItemMessage( hwnd



        }
        // number of words on ch
        number = 1;
        // fill up choice list i
        if( number < 9 )
                for ( int k = nu
                {
                        static W

                        // Get l
                        WordOC::

choicePQ[ k ]->word );

                        // numbe
                        int numb

                        // Add w
                        for( int
                        {
                        if( numb
                        {
                        char wor

                        strcpy(
                                                            if( LB_E
=> RR == SendDlgItemMessage( hwnd,
=>                          CHOICE_LIST_ID,
=>                          LB_FINDSTRING,
=>                          0, (DWORD) wordSpelling) )

=> SendDlgItemMessage( hwnd,
=>                          CHOICE_LIST_ID,
=>                          LB_ADDSTRING,
=>                          0, (DWORD) wordSpelling);
                                                            } // if(
=>  number < 9 )
                                                            else
=> break;
                                                            number++
=> ;
                                                      } // for number0
=> fWords
                                                      if( number >= 9
=> )
                                                      {
                                                            tmpVoc->
=> removeAll();
                                                            break;
                                                      }
                                                      } // for ( k )
                                          }     // recog->recog()
                                          else
                                                SendDlgItemMessage( hwnd
=> , CHOICE_LIST_ID,
=>                 LB_ADDSTRING,
=>                 0,
=>                 (DWORD ) "no recognition");
                                                recog->setChannel( chan );
                                                numWords = 0;
                                          }// if (numWords)

                                          // reset to 15
                                          speechTask->setPar( "computation", (shor
=> t )15 );
                                          // class recognizer knows about the retu
=> rned choice list,
                                          // built up in decreasing confidence ord
=> er.
                                          addressConfidence = recog->confidence();
```

CHOICE.CPP 10-26-95 3:18p

```
                                          // If the confidence is lower than 4, we
=>  do not trust answer
                                          if( addressConfidence < 4 )
                                          {
                                                itsBad( hwnd, &alphaUtt );
                                                break;
                                          }

                                          // clean up temp state
                                          recog->deleteState( tmpState );

                                          // bingo: display our best guess for let
=> ters
                                          SendDlgItemMessage( hwnd, CHOICE_LIST_ID
=> , LB_ADDSTRING, 0, (DWORD ) alphaSpelling);

                                          appState = WORD_STATE;

                                    } // if (recog->recog()) on alpha voc

                                    // else recog failed!!!

                                    // clean up, change back to alpha voc where phra
=> ses were built
                                    recog->setVoc( alphaVoc );

                                    // Remove the 40 hypotheses in the zipcommandwor
=> ds state
                                    recog->setState( "zipcommandwords" );

                                    alphaHypoAC.cleanUpNewWords( recog );

                                    recog->killUtt();
                              } // case ALPHA_STATE

                              break;

                              //·······················································
=> ·······
                              // Now we are out of the message loop. Our app is waitin
=> g
                              // for a new message from the FEP for speech. The next t
=> ime
                              // speech is heard we will be in the state CITY_STATE.


                              // We are sleeping, and expecting to wake up.
                              // Re initialize text on screen.

                        case WAKEUP_STATE:
                              {
                                    SetWindowText( GetDlgItem( hwnd, CHOICE_LIST_ID
=> ), "Awake" );

                                    recog->beep( "awake.wav" );

                                    SetWindowText( GetDlgItem( hwnd, ALPHA_CHOICE_LI
=> ST), "" );

                                    appState = ALPHA_STATE;

                                    recog->killUtt();
                              }
                              break;
                  }
EndOfSwitch:
```

Page 11 of 17

```
                // we always get to this end switch
                SetWindowText( GetDlgItem( hwnd, CONFIDENCE_TEXT_ID ), itoa( add
 => ressConfidence, buf, 10 ) );

                SetWindowText( GetDlgItem( hwnd, PROMPT_TEXT_ID ), sayPrompt[ ap
 => pState ] );

                // In case there is still another utterance, but we have no star
 => t of
                // utterance message, put a start message in the queue.
                if( recog->peekUtt() )
                {
                        postSpeechEvent( (SD_CHANNEL )1, SD_CHANNEL_START );
                }
        }
}

void DlgWin::onSysCommand( HWND hwnd, UINT cmd, int, int )
{
        switch( cmd )
        {
         case SC_CLOSE:
                EndDialog( hwnd, 0 );
                break;
        }
}

// Function that organizes the user interface

void DlgWin::onCommand( HWND hwnd, UINT cmd, HWND, UINT )
{
        switch ( cmd )
        {
         case CHOICE_LIST_ID:
                break;

        case ALPHA_CHOICE_LIST:
                break;

         case USR_COMBO_ID:
                break;

         case SAVE_BTN_ID:
                if( recog && channel && recog->getUser() )
                {
                        channel->flush();

                        recog->saveUser( origCityUser );
                        cityUser = origCityUser;
                }
                break;


         case EXIT_BTN_ID:
                EndDialog( hwnd, 0 );
                break;


         case TRAIN_BTN_ID:
                // the function lpfnTrain has been initialized during processing
                // of WM_CREATE in AppWin::onCreate
                if (cityUser != 0)
                        recog->closeUser( cityUser );
```

```
                DialogBox( hInstance, "TrainDialog", hWnd, (DLGPROC) lpfnTrain )
=> ;
            FreeProcInstance( lpfnTrain );

                break;


        case PURGE_BTN_ID:
                // Get rid of previously assigned memory blocks
                if( speechTask )
                {
                        HCURSOR hCursor = SetCursor( LoadCursor( 0, IDC_WAIT ) )
=> ;

                        alphaUtt.killUtt();

                        int hadListened =  channel != 0 && recog != 0;

                        int wasListening = hadListened && channel->isListening()
=> ;

                        if( wasListening )
                                channel->listen( FALSE );

                        if( hadListened )
                        {
                                recog->saveUser( tmpCityUsr );
                                cityUser = tmpCityUsr;
                        }

                        if( channel )
                        {
                                delete channel;
                                channel = 0;
                        }

                        if( recog )
                        {
                                delete recog;
                                recog = 0;
                        }
                        delete speechTask;

                        speechTask = new SpeechTask( name, reportError );

                        if( hadListened )
                        {
                                channel = new WindowsLiveChannel( "init.txt" );

                                recog = new Recognizer( channel );

                                if(
                                        recog->setUser( cityUser ) == 0 ||
                                        recog->setVoc( alphaVoc ) == 0 ||
                                        recog->setVoc( cityVoc ) == 0 )
                                {
                                        MessageBox( hwnd, "Could not open either
=>  alfa.voc or ctall.voc or user file!",
=>         DlgWin::name, MB_ICONEXCLAMATION | MB_OK );

                                        EndDialog( hwnd, 0 );
                                }

                                // handle the callback function to the FEP
                        // Look in dragcpp.cpp for more details how the
=> recognizer
                        // class deals with all this
                        channel->open( postSpeechEvent );

                        if( wasListening )
                                channel->listen( TRUE );
                }

                SetCursor( hCursor );
        }

        break;

    case SETUSER_BTN_ID:
        {
                char usrName[ UTT_PROMPT_LENGTH ];

                if( !GetDlgItemText( hwnd, USR_COMBO_ID, usrName, sizeof
=> (usrName) ) )
                        break;

                HCURSOR hCursor = SetCursor( LoadCursor( 0, IDC_WAIT ) )
=> ;

                bool isMale = tolower( usrName[ strlen( usrName ) - 1 ]
=> ) == 'm';

                if( recog )
                {
                        extern int fileNameCmp( const char *, const char
=> * );

                        char buffer[ 128 ];
                        recog->getUsername( buffer, sizeof( buffer ) );

                        if( fileNameCmp( usrName, buffer ) )
                        {
                                bool wasListening = channel->isListening
=> ();

                                // Turn mic off temporarily
                                if( wasListening )
                                        channel->listen( FALSE );

                                recog->closeVoc( cityVoc );
                                recog->closeVoc( alphaVoc );
                                recog->closeUser( cityUser );

                                cityUser = origCityUser;

                                strcpy( cityUser, usrName );

                                if( isMale )
                                {
                                        alphaVoc = malalphaVoc;
                                        cityVoc = malCityVoc;
                                }
                                else
                                {
                                        alphaVoc = femalphaVoc;
                                        cityVoc = femCityVoc;
                                }

                                if(
                                         recog->setUser( cityUser ) == 0
```

CHOICE.CPP 10-26-95 3:18p

```
=> ||
                                         recog->setVoc( alphaVoc ) == 0
=> ||
                                         recog->setVoc( cityVoc ) == 0 )
                                {
                                        MessageBox( hwnd, "Could not ope
=> n either alfa.voc or ctall.voc or user file!",
=>                      DlgWin::name, MB_ICONEXCLAMATION | MB_OK );

                                        EndDialog( hwnd, 0 );
                                }
                                // reset to on after loading
                                if( wasListening )
                                        channel->listen( TRUE );
                        }
                }
                else    // recog == 0
                {
                        // Load male vocabulary
                        if( isMale )
                        {
                                alphaVoc = malalphaVoc;
                                cityVoc = malCityVoc;
                        }

                        // Load female vocabulary
                        else
                        {
                                alphaVoc = femalphaVoc;
                                cityVoc = femCityVoc;
                        }

                        // Get the channel
                        if( channel == 0 )
                                channel = new WindowsLiveChannel("init.t
=> xt");

                        // Get the channel
                        if( channel == 0 )
                                channel = new WindowsLiveChannel(usrName
=> );

                        // define new recognizer
                        recog = new Recognizer( channel );

                        strcpy( cityUser, usrName );

                        if(
                                 recog->setUser( cityUser ) == 0 ||
                                 recog->setVoc( alphaVoc ) == 0 ||
                                 recog->setVoc( cityVoc ) == 0 )
                        {
                                MessageBox( hwnd, "Could not open either
=>  digits.voc or ctall.voc or user file!",
=>          DlgWin::name, MB_ICONEXCLAMATION | MB_OK );

                                EndDialog( hwnd, 0 );
                        }

                        channel->open( postSpeechEvent );

#if 0
                        // set parameter to collect wave forms
                        speechTask->setPar( "collect-waveform", (short )
=>  TRUE );

                        speechTask->setPar( "save-waveform", (short ) TR
```

Page 13 of 17

CHOICE.CPP 10-26-95 3:18p

```
  => UE );
#endif
                                EnableWindow( GetDlgItem( hwnd, MICROPHONE_BTN_I
  => 0 ), TRUE );

                                //              EnableWindow( GetDlgItem( hwnd,
  => SETUSER_BTN_ID ), FALSE );

                                SetFocus( GetDlgItem( hwnd, WORD_PAUSE_SLIDE_ID
  => ) );

                                numErrors = 0;

                                PostMessage( hwnd, WM_COMMAND, WORD_PAUSE_SLIDE_
  => ID, 0 );
                                PostMessage( hwnd, WM_COMMAND, START_THRESH_SLID
  => E_ID, 0 );
                                PostMessage( hwnd, WM_COMMAND, END_THRESH_SLIDE_
  => ID, 0 );
                        }

                        SetCursor( hCursor );
                }
                break;

        case MICROPHONE_BTN_ID:
                // if there is a recognizer
                if( recog )
                {
                        char num[ 20 ];
                        ltoa( computeMemoryUse(), num, 10 );
                        SetWindowText( GetDlgItem( hwnd, MEMORY_TEXT_ID ), num )
  => ;

                        // boolean behaviour ON or OFF
                        recog->listen( !recog->isListening() );

                        SetWindowText( GetDlgItem( hwnd, MICROPHONE_BTN_ID ),
  =>  recog->isListening() ? "&Microphone On" : "&Microphone Off" );

                        // When microphone is turned off
                        if( !recog->isListening() )
                        {
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_RES
  => ETCONTENT, 0, 0 );

                                long total = right + wrong;
                                if( total == 0 ) total = 1;

                                sprintf( buf, "right:    %ld  %ld\%", right, (10
  => 0*right)/total );
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADD
  => STRING, 0, (DWORD )buf );
                                sprintf( buf, "wrong:    %ld  %ld\%", wrong, (10
  => 0*wrong)/total );
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADD
  => STRING, 0, (DWORD )buf );
                                sprintf( buf, "predicted: %ld  %ld\%", predicted
  => , (100*predicted)/total );
                                SendDlgItemMessage( hwnd, CHOICE_LIST_ID, LB_ADD
  => STRING, 0, (DWORD )buf );

                                KillTimer( hwnd, VUTIMER_ID );
                        }
```

Page 14 of 17

```
                        // mic is listening
                        else
                        {
                                SetTimer( hwnd, VUTIMER_ID, 20, (TIMERPROC )0 );
                        }

                }
                break;

        case WORD_PAUSE_SLIDE_ID:
                {
                        wordPause = GetScrollPos( GetDlgItem( hwnd, WORD_PAUSE_S
=> LIDE_ID ), SB_CTL );

                        bool wasListening = ( recog != 0 && recog->isListening()
=>  );

                        if( wasListening )      recog->listen( 0 );

                        if( speechTask )        speechTask->setPar( "word-pause"
=> , (short )wordPause );

                        if( wasListening )      recog->listen( 1 );

                        char num[ 20 ];

                        itoa( wordPause, num, 10 );

                        SetWindowText( GetDlgItem( hwnd, WORD_PAUSE_TEXT_ID ), n
=> um );
                }
                break;

        case START_THRESH_SLIDE_ID:
                {
                        startThresh = GetScrollPos( GetDlgItem( hwnd, START_THRE
=> SH_SLIDE_ID ), SB_CTL );

                        bool wasListening = ( recog != 0 && recog->isListening()
=>  );

                        if( wasListening )      recog->listen( 0 );

                        if( speechTask )        fep_Setpar( SUPER_USER, FEP_STAR
=> TOFSPEECHTHRESH,

=>                                                                    (WORD ) sta
=> rtThresh );

                        if( wasListening )      recog->listen( 1 );

                        char num[ 20 ];

                        itoa( startThresh, num, 10 );

                        SetWindowText( GetDlgItem( hwnd, START_THRESH_TEXT_ID ),
=>  num );
                }
                break;

        case END_THRESH_SLIDE_ID:
                {
                        endThresh = GetScrollPos( GetDlgItem( hwnd, END_THRESH_S
=> LIDE_ID ), SB_CTL );

                        bool wasListening = ( recog != 0 && recog->isListening()
=>  );
                        if( wasListening )      recog->listen( 0 );

                        if( speechTask )        Fep_Setpar( SUPER_USER, FEP_ENDO
=> FOSPEECHTHRESH,
                                                                    (WORD ) end
=>
=> Thresh );

                        if( wasListening )      recog->listen( 1 );

                        char num[ 20 ];

                        itoa( endThresh, num, 10 );

                        SetWindowText( GetDlgItem( hwnd, END_THRESH_TEXT_ID ), n
=> um );
                }
                break;
        }
}

void DlgWin::onTimerDrawVU( HWND hwnd, UINT id )
{
        if( id == VUTIMER_ID && channel && channel->isListening() )
        {
                bool test;

                FEP_S_STATUS fepinfo;

                Fep_Status( (BOOL* )&test, &fepinfo ) ;

                HWND hvu = GetDlgItem( hwnd, VUMETER_ID );

                RECT rc;

                GetClientRect( hvu, &rc );

                //      rc.left     = 95;
                //      rc.right    = 95 + 61;
                //      rc.top      = 116;
                //      rc.bottom   = 116 + 3;

                HDC hdc = GetDC( hvu );

                int noiseright, speechleft, speechright, unit;

                unit = (rc.right - rc.left) / fepinfo.max_level;

                noiseright = speechleft = unit * fepinfo.noise_level;

                speechright = unit * fepinfo.speech_level + speechleft;

                //      draw speech
                HBRUSH hBrush = CreateSolidBrush( RGB( 255, 0, 0 ) );

                HBRUSH hOldBrush = (HBRUSH )SelectObject( hdc, hBrush );

                Rectangle( hdc, rc.left, rc.top, noiseright, rc.bottom );

                SelectObject( hdc, hOldBrush );

                DeleteObject(hBrush);

                        CHOICE.CPP   10-26-95   3:18p
```

```
                //      draw speech
                hBrush = CreateSolidBrush( RGB( 0, 255, 0 ) );

                hOldBrush = (HBRUSH )SelectObject( hdc, hBrush );

                Rectangle( hdc, speechleft, rc.top, speechright, rc.bottom );

                SelectObject( hdc, hOldBrush );

                DeleteObject( hBrush );

                // Background
                Rectangle( hdc, speechright, rc.top , rc.right, rc.bottom );

                ReleaseDC( hvu, hdc );
        }
}

void DlgWin::onDestroy( HWND hwnd )
{
        KillTimer( hwnd, VUTIMER_ID );

        if( speechTask )
        {
                wordUtt.killUtt();

                if( channel )
                {
                        delete channel;
                        channel = 0;
                }

                if( recog )
                {
                        delete recog;
                        recog = 0;

#if 0
                        // Print recorded stats on letter strings to file
                        choiceListStats.printStats();
#endif
                }

                delete speechTask;
                speechTask = 0;
        }

        //    PostQuitMessage( 0 );
}


BOOL FAR PASCAL _export DlgWin::wndProc( HWND hwnd, UINT message, WPARAM wParam,
  =>  LPARAM lParam )
{
        messageHandled = TRUE;

        long returnValue = 0;

        switch( message )
        {
                handleMessage( WM_INITDIALOG, hwnd, wParam, lParam, onInitDialog
  =>  );
                handleMessage( WM_DESTROY, hwnd, wParam, lParam, onDestroy );
                                Page 15 of 17
```

```
                              CHOICE.CPP   10-26-95   3:18p
                    handleMessage( WM_COMMAND, hwnd, wParam, lParam, onCommand );
                    handleMessage( WM_SYSCOMMAND, hwnd, wParam, lParam, onSysCommand
 => );
                    handleMessage( WM_TIMER, hwnd, wParam, lParam, onTimerDrawVU );
                    handleMessage( UM_CHANNELSTART, hwnd, wParam, lParam, onSpeech )
 => ;
          }

        return messageHandled ? returnValue : 0;
}


/////////////////////////////////////////////////////////////////////////
//
// CLASS AppWin
//
// Purpose : application driven from this class
//                  WinMain will call wndProc of Appwin
//
//
/////////////////////////////////////////////////////////////////////////

class AppWin
{
 public:
        static long FAR PASCAL _export wndProc( HWND, UINT, WPARAM, LPARAM );

        static HWND      hWnd;
        static HTASK hTask;

 protected:

        static bool messageHandled;

        // wndProc() services
        static bool onCreate( HWND hWnd, CREATESTRUCT FAR* lpCreateStruct );
        static void onDestroy( HWND hWnd );
        static void onNCDestroy( HWND hWnd );
        static void onClose( HWND hWnd );
        static void onCommand( HWND hWnd, UINT cmd, HWND x, UINT y);


        friend int PASCAL WinMain( HINSTANCE, HINSTANCE, LPSTR, int );
};


// Static Global Variables of AppWin
HWND            AppWin::hWnd;
HTASK           AppWin::hTask;
bool            AppWin::messageHandled;


//////////////////////////////////////////////////////////////////////////
// Main function of application
//////////////////////////////////////////////////////////////////////////


int PASCAL WinMain( HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR lpszCmdLine,
 =>  int /* nCmdShow */ )
{
        static const char name[] = "Barbados spell mode";

        SetMessageQueue( 64 );
```

```
                                                          Page 16 of 17
     if( !hPrevInstance )
     {
            WNDCLASS wndclass;

            wndclass.style          = CS_HREDRAW | CS_VREDRAW;
            wndclass.lpfnWndProc    = AppWin::wndProc;
            wndclass.cbClsExtra     = 0;
            wndclass.cbWndExtra     = 0;
            wndclass.hInstance      = hInst;
            wndclass.hIcon          = LoadIcon( 0, MAKEINTRESOURCE( IDI_MY_IC
=> ON ) );
            wndclass.hCursor        = LoadCursor( 0, IDC_ARROW );
            // wndclass.hIcon          = LoadIcon( hInst, "merc.ico" );
            // wndclass.hCursor        = LoadCursor( 0, IDC_ARROW );
            wndclass.hbrBackground = 0;
            wndclass.lpszMenuName  = 0;
            wndclass.lpszClassName = name;

            RegisterClass( &wndclass );
     }

     ::hInstance = hInst;    // the global hInstance for the sliderctl
     slideControl.libMainInit();     // register the slider class

     HWND hWnd = CreateWindow( name,
                                                                   name,
                                                                   WS_OVERLAPPEDWIN
=> DOW | WS_CLIPCHILDREN,
                                                                   CW_USEDEFAULT,
                                                                   CW_USEDEFAULT,
                                                                   CW_USEDEFAULT,
                                                                   CW_USEDEFAULT,
                                                                   0,
                                                                   0,
                                                                   hInst,
                                                                   lpszCmdLine );


     assert( hWnd == AppWin::hWnd );

     ShowWindow( AppWin::hWnd, SW_HIDE );
     //     ShowWindow( hWnd, nCmdShow );
     //     UpdateWindow( hWnd );

     MSG msg;

     while( GetMessage( &msg, 0, 0, 0 ) )
     {
            TranslateMessage( &msg );
            DispatchMessage( &msg );
     }

     return msg.wParam;
}


// Application Helper function

void AppWin::onCommand( HWND hWnd, UINT cmd, HWND, UINT )
{
     switch ( cmd )
     {
      case UM_INIT_DIALOG:
            {
                    // create dialog box proc instances for use later
```

```
                        static FARPROC lpfn = MakeProcInstance( (FARPROC )DlgWin
  => ::wndProc, hInstance );

                        assert( lpfn );

                        DialogBox( hInstance, "ChoiceDialog", 0, (DLGPROC )lpfn
  => );

                        PostMessage( hWnd, WM_CLOSE, 0, 0 );
                }
                break ;


         default:
                messageHandled = FALSE;
        }
}

// Application Helper function

bool AppWin::onCreate( HWND hwnd, CREATESTRUCT FAR* )
{
        hTask = GetCurrentTask();

        hWnd = hwnd;

        PostMessage( hwnd, WM_COMMAND, UM_INIT_DIALOG, 0 );

        return TRUE;
}

// Application Helper function

void AppWin::onDestroy( HWND )
{
        PostQuitMessage( 0 );
}


// Application Helper function

void AppWin::onNCDestroy( HWND )
{
}


// Application Helper function

void AppWin::onClose( HWND hWnd )
{
        DestroyWindow( hWnd );
}


// Application message handler called from winMain function

long FAR PASCAL _export AppWin::wndProc( HWND hwnd, UINT message, WPARAM wParam,
  =>  LPARAM lParam )
{
        messageHandled = TRUE;
```

```
        long returnValue = 0;

        switch( message )
        {
                handleMessage( WM_CREATE, hwnd, wParam, lParam, onCreate );
                handleMessage( WM_DESTROY, hwnd, wParam, lParam, onDestroy );
                handleMessage( WM_NCDESTROY, hwnd, wParam, lParam, onNCDestroy )
  => ;
                handleMessage( WM_CLOSE, hwnd, wParam, lParam, onClose );
                handleMessage( WM_COMMAND, hwnd, wParam, lParam, onCommand );

         default:
                messageHandled = FALSE;

                break;
        }

        return( messageHandled
                                        ? returnValue
                                        : DefWindowProc( hwnd, message, wParam,
  => lParam ) );
}
```

## Claims

1. A method of speech recognition including:

   recognizing a first utterance.
   recognizing a second utterance having information that is related to said first utterance. and
   determining the most probable first and second utterances based on stored information about valid relationships between possible first and second utterances.

2. The method of claim 1 further including determining validity of one of the recognized utterances and including an invalid utterance in a list of possible utterances for comparison with possible utterances in said list.

3. The method of claim 1 wherein determining the most probable utterances includes rerecognition of one of said utterances against a list of possible utterances.

4. The method of claim 3 further including ranking said list of possible utterances based upon how closely each of the possible utterances corresponds to said one of said utterances.

5. The method of claim 1 further including creating an hypothesized list of possible utterances that relate to at least one of said recognized utterances based on said stored information.

6. The method of claim 5 further including ranking a list of possible utterances based upon how closely each of the possible utterances corresponds to one of said utterances and comparing said ranked list of possible utterances to said hypothesized list of possible utterances for commonality between said lists.

7. The method of claim 1 further including creating an hypothesized list of possible utterances that relate to at least one of said recognized utterances based on said stored information.

8. The method of claim 1 further including creating a list of possible utterances that could be confused with at least one of said recognized utterances.

9. The method of claim 1 wherein said recognized first utterance is recognized continuously.

10. The method of claim 1 wherein said recognized second utterance is recognized discretely.

11. The method of claim 9 further including creating a list of possible first utterances that may be confused with said recognized first utterance.

12. The method of claim 11 further including creating an hypothesized list of possible second utterances that relate to said possible first utterances based on said stored information.

13. The method of claim 12 further including adding said recognized second utterance to said hypothesized list of possible second utterances to create a merged list of possible second utterances.

14. The method of claim 13 further including rerecognizing said merged list of possible second utterances against said second utterance to get a ranked list of possible second utterances, said ranking based upon how closely each possible second utterance in said merged list corresponds to said second utterance.

15. The method of claim 14 further including comparing said ranked list of possible second utterances to said hypothesized list of possible second utterances for commonality between said lists. said highest ranked possible second utterance in said ranked list being compared first.

16. The method of claim 15 further including creating an hypothesized list of possible first utterances from said second recognized utterance based on said stored information.

17. The method of claim 16 further including adding said recognized first utterance to said hypothesized list of possible first utterances to create a merged list of possible first utterances.

18. The method of claim 17 further including rerecognizing said merged list of possible first utterances against said first utterance to get a ranked list of possible first utterances having a ranking based upon how closely each possible first utterance in said merged list corresponds to said first utterance.

19. The method of claim 18 further including evaluating a possible first utterance ranked second in said ranked list of possible first utterances by determining whether a distance parameter associated with said second ranked possible first utterance is within an acceptable limit.

20. The method of claim 19 further including indicating to a user when said second ranked possible first utterance is not within said acceptable limit and no commonality exists between said ranked list of possible second utterances and said hypothesized list of possible second utterances.

21. The method of claim 1 wherein said first utterance comprises a zipstate and said second utterance comprises a city from a destination address on a package, said determination of the most probable first and second utterances resulting in the sorting of said package according to the package's destination address.

22. The method of claim 1 wherein said first utterance comprises a word and said second utterance comprises spelled prefix including ordered symbols.

23. The method of claim 22 further including creating a list of possible prefixes that could be confused with said recognized prefix.

24. The method of claim 23 wherein creating said list of possible prefixes includes determining, in the context of a preceding symbol or silence, a probability of confusing each recognized symbol in said prefix with each symbol in a list of possible symbols.

25. The method of claim 23 wherein creating said list of possible prefixes includes determining, in the context of a preceding symbol or silence, a probability of confusing each recognized symbol in said prefix with more than one symbol.

26. The method of claim 23 wherein creating said list of possible prefixes includes determining, in the context of a preceding symbol or silence, a probability of confusing each recognized symbol in said prefix with an absence of a symbol.

27. The method of claim 23 wherein creating said list of possible prefixes includes replacing a sequence of symbols with a single symbol.

28. The method of claim 22 wherein said first utterance comprises a spelled word and said determination of the most probable first and second utterances results in recognizing said spelled word.

29. A method of generating a choice list from a continuously recognized utterance comprising:

    recognizing a spoken utterance,
    consulting stored information to determine the probability of confusing possible utterances in the stored information with said recognized utterance, and
    producing a list of possible utterances from the stored information that could be confused with the recognized utterance.

30. The method of claim 29 further including rerecognizing said utterance against a merged list of said list of possible utterances and said recognized utterance to create a ranked list of possible utterances having a ranking based upon how closely each utterance in said merged list corresponds to said spoken utterance.

31. A method of recognizing ambiguous inputs including:

    recognizing a first ambiguous input,
    recognizing a second ambiguous input having information that is related to said first ambiguous input, and
    determining the most probable first and second ambiguous inputs based on stored information about valid relationships between possible first and second ambiguous inputs.

**32.** A method of training a speech recognizer, comprising:

prompting a user to make a first utterance comprising symbols.
recognizing said symbols, and
calculating the probability of confusing each recognized symbol with the prompted symbol.

**33.** The method of claim 32 wherein said probabilities are calculated within the context of the preceding symbol or silence.

**34.** A method of displaying word choices during speech recognition, comprising:

recognizing an uttered word,
recognizing a spelling of a prefix of the word, whereby symbols are used to spell said prefix, and
displaying a list of word choices on a screen for selection, a top choice on the list corresponding to a highest ranked choice.

**35.** The method of claim 34 wherein said symbols comprise letters, digits, and punctuation.

**36.** The method of claim 29 wherein said spoken work is recognized continuously.

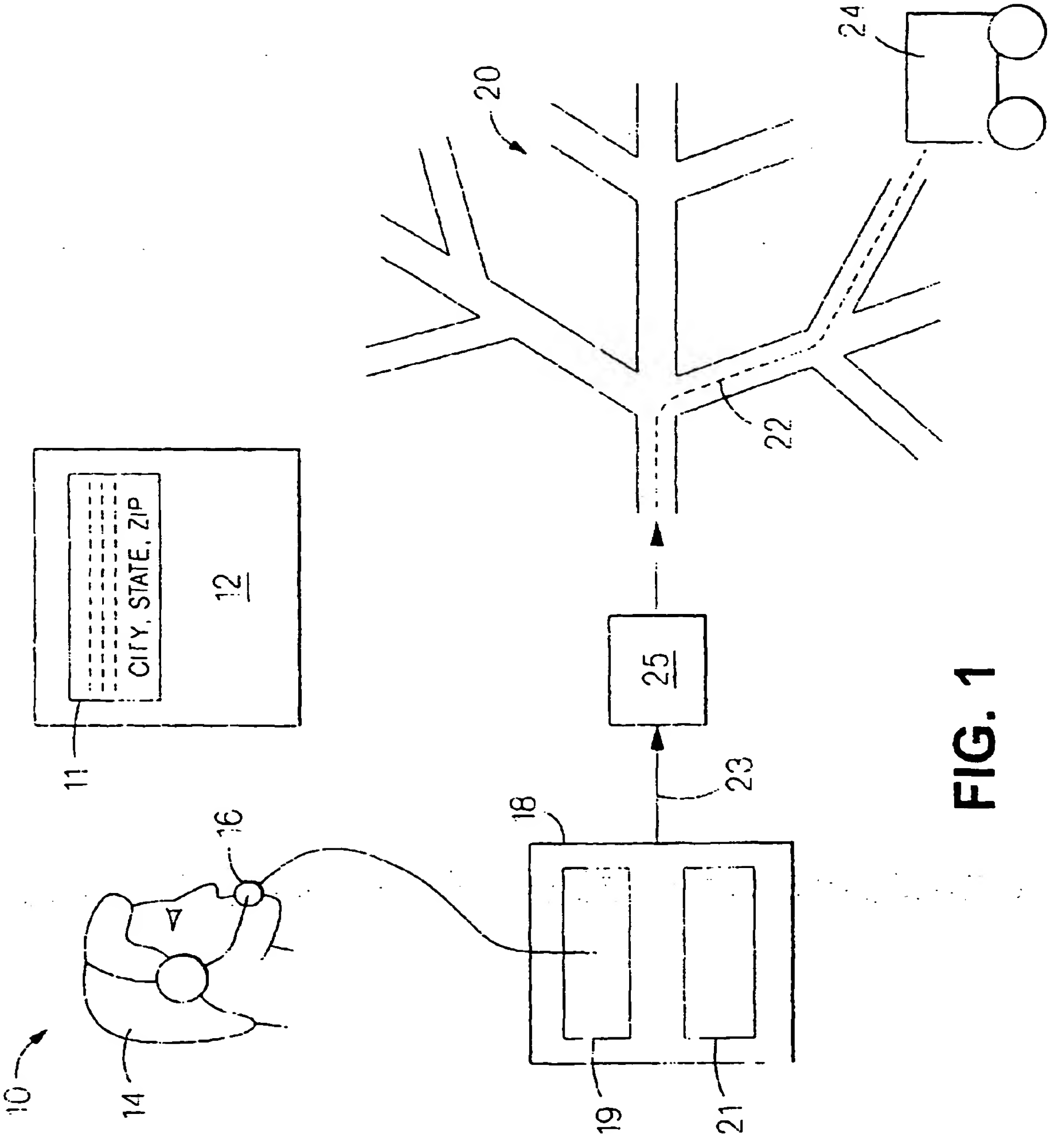

10
14
16
11
CITY, STATE, ZIP
12
18
19
21
23
25
20
22
24

FIG. 1

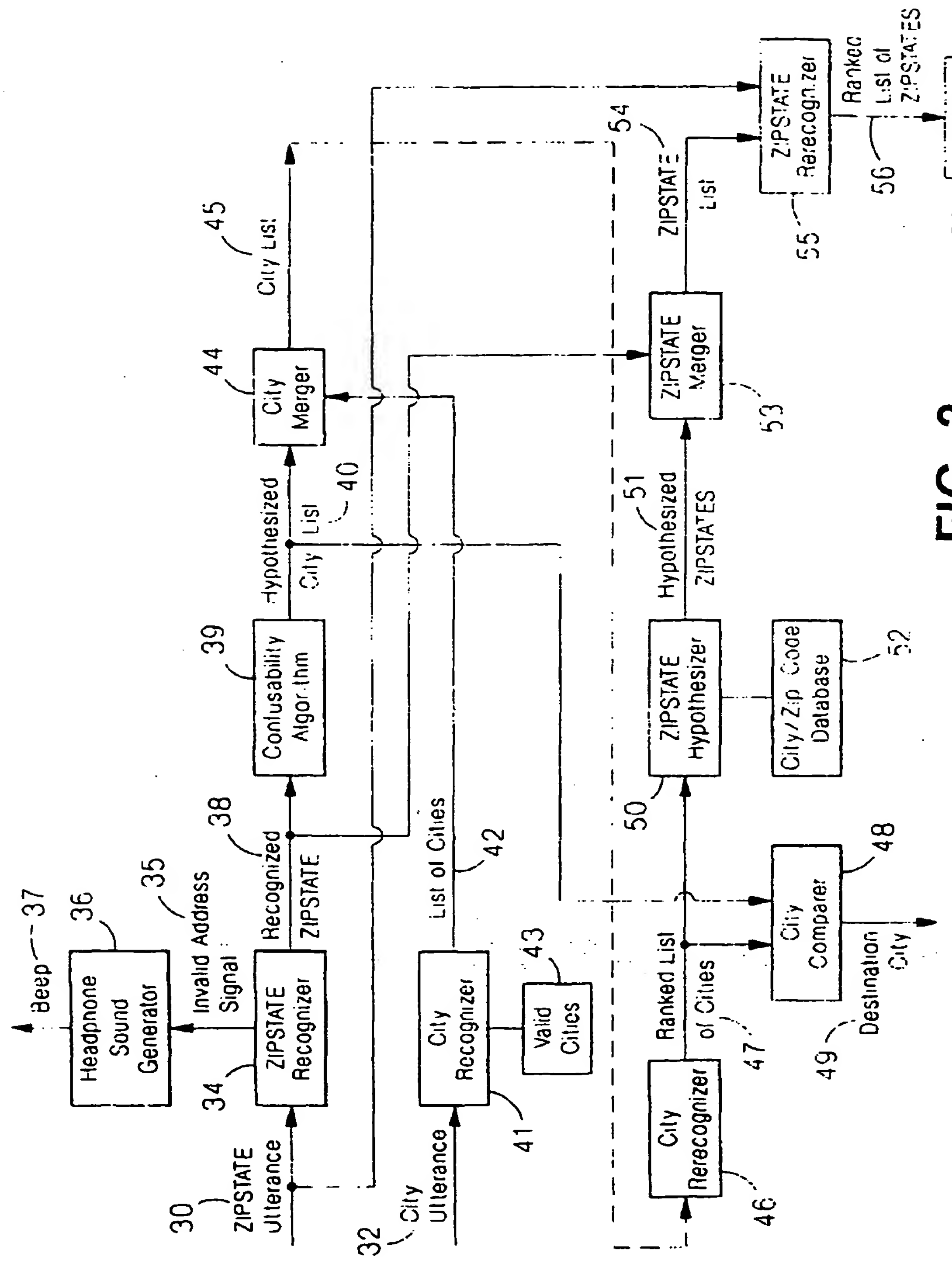

30
ZIPSTATE Utterance
34
ZIPSTATE Recognizer
Invalid Address Signal
35
36
Headphone Sound Generator
Beep
37
Recognized ZIPSTATE
38
Confusability Algorithm
39
Hypothesized City List
40
City Merger
44
City List
45
32
City Utterance
City Recognizer
41
Valid Cities
43
List of Cities
42
City Rerecognizer
46
Ranked List of Cities
47
City Comparer
48
Destination City
49
ZIPSTATE Hypothesizer
50
City / Zip Code Database
52
Hypothesized ZIPSTATES
51
ZIPSTATE Merger
53
ZIPSTATE List
54
ZIPSTATE Rerecognizer
55
Ranked List of ZIPSTATES
56
ZIPSTATE Evaluator
57

FIG. 2

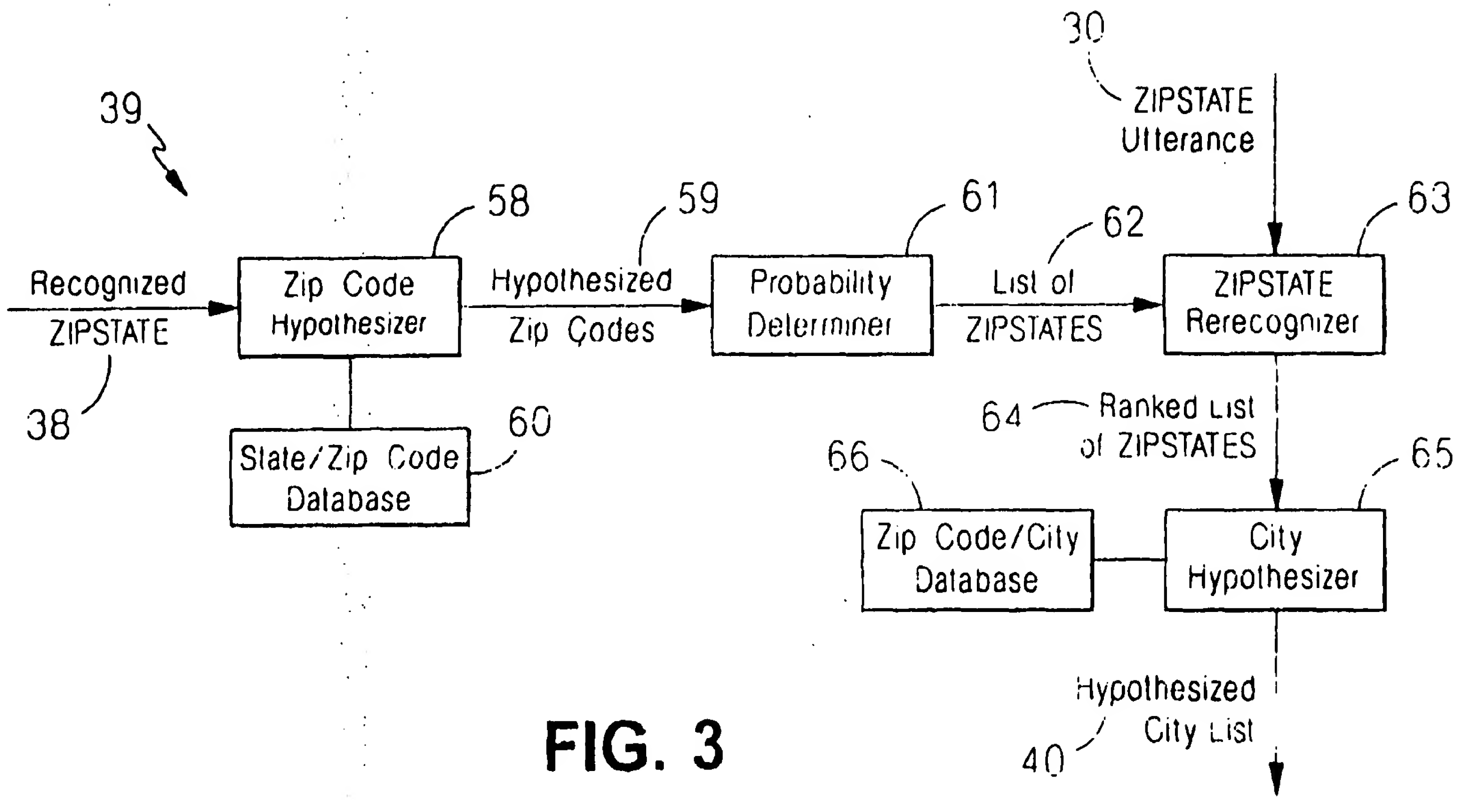

39
Recognized ZIPSTATE
38
Zip Code Hypothesizer
58
State/Zip Code Database
60
Hypothesized Zip Codes
59
Probability Determiner
61
List of ZIPSTATES
62
30
ZIPSTATE Utterance
ZIPSTATE Rerecognizer
63
64
Ranked List of ZIPSTATES
66
Zip Code/City Database
City Hypothesizer
65
Hypothesized City List
40

FIG. 3

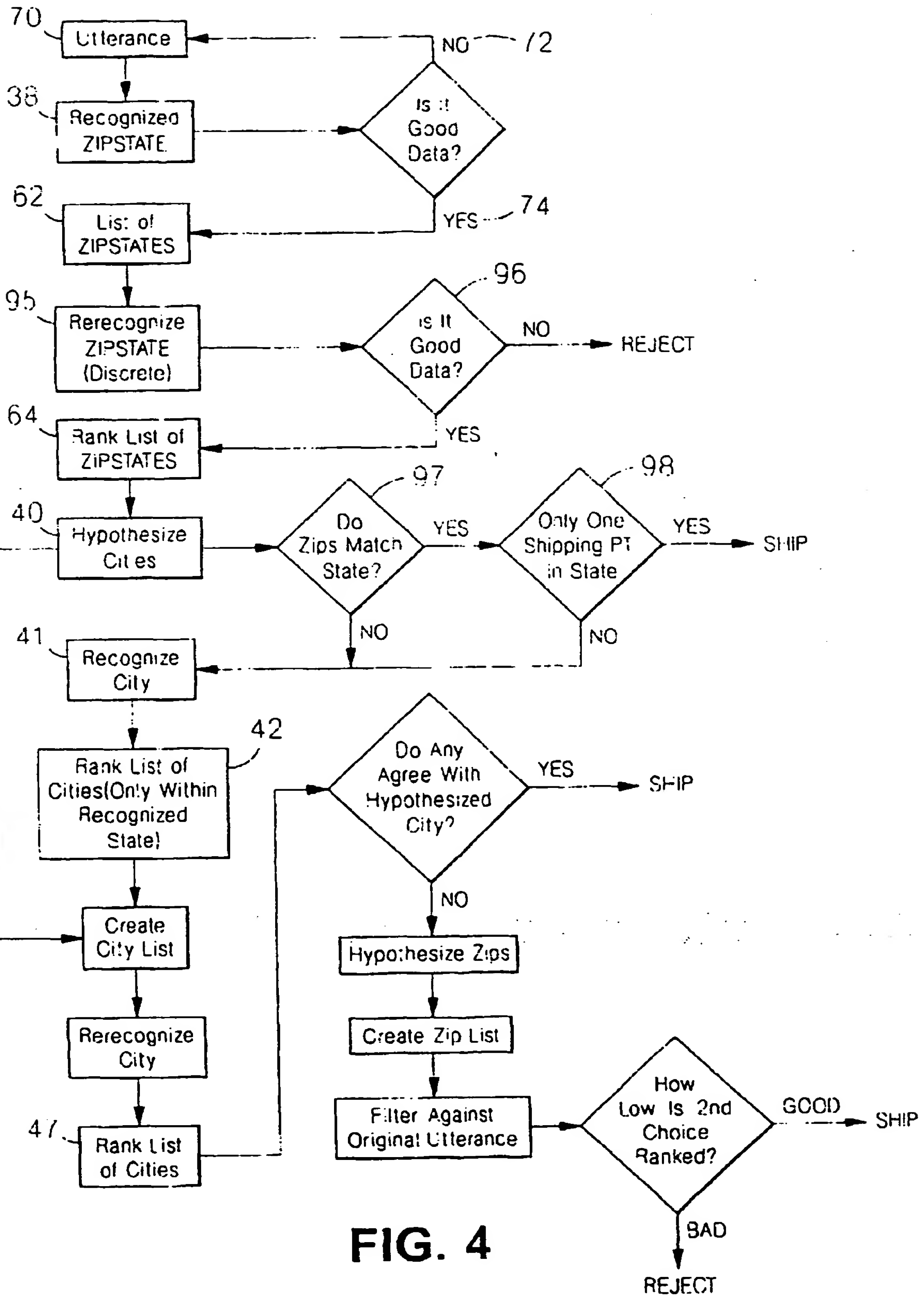

70
Utterance
NO
72
38
Recognized ZIPSTATE
Is It Good Data?
YES
74
62
List of ZIPSTATES
96
95
Rerecognize ZIPSTATE (Discrete)
Is It Good Data?
NO
REJECT
YES
64
Rank List of ZIPSTATES
97
98
40
Hypothesize Cities
Do Zips Match State?
YES
Only One Shipping PT in State
YES
SHIP
NO
NO
41
Recognize City
42
Rank List of Cities(Only Within Recognized State)
Do Any Agree With Hypothesized City?
YES
SHIP
NO
Create City List
Hypothesize Zips
Create Zip List
Rerecognize City
47
Rank List of Cities
Filter Against Original Utterance
How Low Is 2nd Choice Ranked?
GOOD
SHIP
BAD
REJECT

FIG. 4

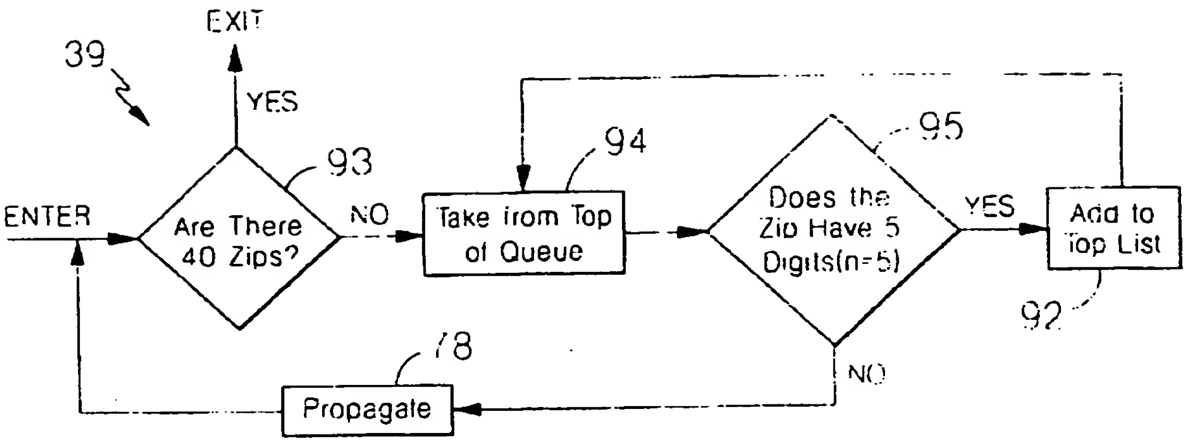

EXIT
39
YES
93
ENTER
Are There 40 Zips?
NO
94
Take from Top of Queue
95
Does the Zip Have 5 Digits(n=5)
YES
Add to Top List
92
NO
78
Propagate

FIG. 4a

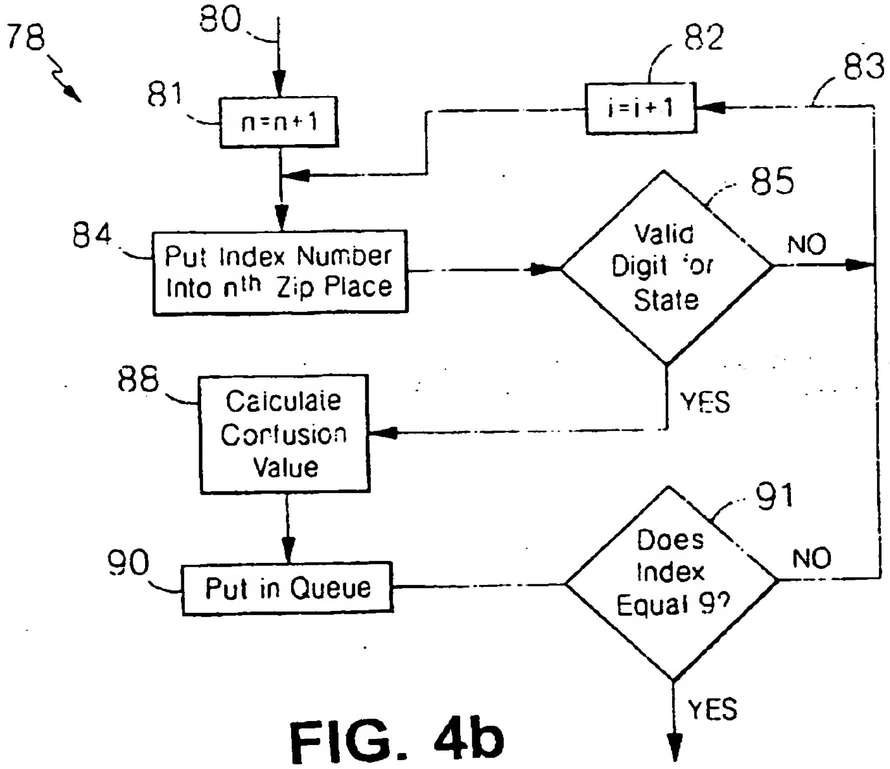

78
80
81
n=n+1
82
i=i+1
83
84
Put Index Number Into n^th Zip Place
85
Valid Digit for State
NO
YES
88
Calculate Confusion Value
91
90
Put in Queue
Does Index Equal 9?
NO
YES

FIG. 4b

Confusability Array (of Probabilities)

Hypothesized

| Recognized | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1, | 0.0204, | 0.0497, | 0.0356, | 0.0193, | 0.0216, | 0.0204, | 0.0228, | 0.0001, | 0.0001, |
| 1 | 0.0155, | 1, | 0.0001, | 0.0001, | 0.0917, | 0.0734, | 0.0001, | 0.0421, | 0.0001, | 0.0301, |
| 2 | 0.0734, | 0.0204, | 1, | 0.1888, | 0.0398, | 0.0285, | 0.1211, | 0.0526, | 0.0421, | 0.0301, |
| 3 | 0.0094, | 0.0001, | 0.0497, | 1, | 0.0001, | 0.0285, | 0.0285, | 0.0001, | 0.0588, | 0.0497, |
| 4 | 0.0155, | 0.0917, | 0.0216, | 0.0216, | 1, | 0.0657, | 0.0001, | 0.0588, | 0.0001, | 0.0301, |
| 5 | 0.0094 | 0.0204, | 0.0001, | 0.0301, | 0.0193, | 1, | 0.0001, | 0.0001, | 0.0001, | 0.1353, |
| 6 | 0.0053, | 0.0001, | 0.0285, | 0.0356, | 0.0001, | 0.0001, | 1, | 0.0001 | 0.0001, | 0.0001, |
| 7 | 0.0094, | 0.0001, | 0.0216, | 0.0216, | 0.0193, | 0.0285, | 0.0001, | 1, | 0.0001, | 0.0301, |
| 8 | 0.0155, | 0.0270, | 0.0337, | 0.0374, | 0.0001, | 0.0001, | 0.0285, | 0.0001, | 1, | 0.0001 |
| 9 | 0.0001, | 0.0001, | 0.0216, | 0.0001, | 0.0001 | 0.0285, | 0.0001, | 0.0001, | 0.0228, | 1 |

FIG. 5

|  | 0 | Code | Probability of Confusing Recognized Digit With i | Confusion Value |
|---|---|---|---|---|
| n=1:<br>Recognized<br>Digit=0 | 1 | 0____ | 1 | 1 |
| n=2.<br>Recognized<br>Digit=3 | 1 | 01___ | 0.0001 | 0.0001 |
|  | 2 | 02___ | 0.0497 | 0.0497 |
| n=3<br>Recognized<br>Digit=1 | 0 | 020__ | 0.0155 | (.0497)(.0155)-0.0007 |
|  | 1 | 021__ | 1 | 0.0497 |
|  | 2 | 022__ | 0.0001 | 0 |
|  | 3 | 023__ | 0.0001 | 0 |
|  | 4 | 024__ | 0.0917 | 0.00456 |
|  | 5 | 025__ | 0.0734 | 0.00365 |
|  | 6 | 026__ | 0.0001 | 0 |
|  | 7 | 027__ | 0.0421 | 0.00209 |
| n=4<br>Recognized<br>Digit=1 | 0 | 0210 | 0.0155 | (.0497)(.0155)=0.0007 |
|  | 1 | 0211_ | 1 | 0.0497 |
|  | 2 | 0212_ | 0.0001 | 0 |
|  | 3 | 0213_ | 0.0001 | 0 |
|  | 4 | 0214_ | 0.0917 | 0.00456 |
|  | 5 | 0215_ | 0.0734 | 0.00365 |
|  | 6 | 0216_ | 0.0001 | 0 |
|  | 7 | 0217_ | 0.0421 | 0.00209 |
|  | 8 | 0218_ | 0.0001 | 0 |
|  | 9 | 0219_ | 0.0301 | 0.0015 |
| n=5<br>Recognized<br>Digit=2 | 0 | 02110 | .0734 | (.0497)(.0734)=.00365 |
|  | 1 | 02111 | .0204 | .0010 |
|  | 2 | 02112 | 1 | 0497 |
|  | 3 | 02113 | .1888 | .0094 |
|  | 4 | 02114 | 0398 | .002 |
|  | 5 | 02115 | .0285 | .0014 |
|  | 6 | 02116 | 1211 | .006 |
|  | 7 | 02117 | .0526 | .0026 |
|  | 8 | 02118 | .0421 | .0021 |
|  | 9 | 02119 | .0301 | 0015 |
| n=5 | 0 | 02140 | 0734 | (.00456)(.0734)=.00033 |
|  | 1 | 02141 | 0204 | 0 |
|  | 2 | 02142 | 1 | .00456 |
|  | 3 | 02143 | .1888 | .00086 |
|  | 4 | 02144 | .0398 | .00018 |
|  | 5 | 02145 | .0285 | .00013 |
|  | 6 | 02146 | .1211 | .00055 |
|  | 7 | 02147 | .0526 | .0002 |
|  | 8 | 02148 | .0421 | .00019 |
|  | 9 | 02149 | .0301 | .00014 |

FIG. 6

| Zip Code | Confusion Value |
|---|---|
| 02112 | 0.0497 |
| 02113 | 0.0094 |
| 02116 | 0.00456 |
| 02142 | 0.00456 |
| 02412 | 0.00456 |
| 02110 | 0.00365 |
| 02152 | 0.00365 |
| 02512 | 0.00365 |
| 02117 | 0.0026 |
| 02118 | 0.00209 |
| 02172 | 0.00209 |
| 02712 | 0.00209 |
| 02114 | 0.002 |
| 02119 | 0.0015 |
| 02192 | 0.0015 |
| 02115 | 0.0014 |
| 02111 | 0.001 |
| 02143 | 0.00086 |
| 02413 | 0.00086 |
| 02102 | 0.00077 |
| 02012 | 0.00077 |
| 02153 | 0.00069 |
| 02513 | 0.00069 |
| 02146 | 0.00055 |
| 10173 | 0.00039 |
| 02713 | 0.00039 |
| 02140 | 0.00033 |
| 02542 | 0.00033 |
| 02147 | 0.0002 |
| 02748 | 0.00019 |

FIG. 7

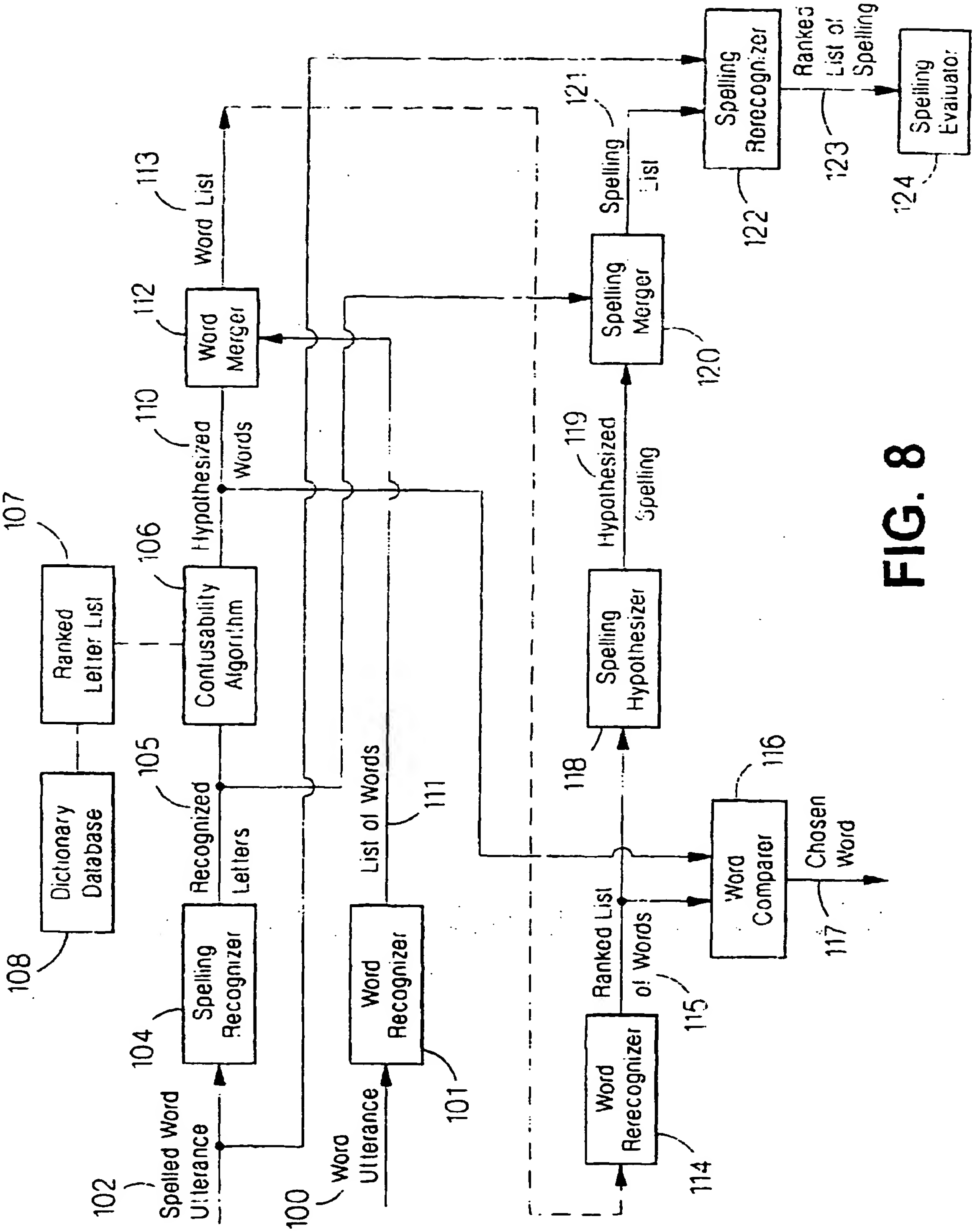

100
Word Utterance
101
Word Recognizer
102
Spelled Word Utterance
104
Spelling Recognizer
105
Recognized Letters
106
Confusability Algorithm
107
Ranked Letter List
108
Dictionary Database
110
Hypothesized Words
111
List of Words
112
Word Merger
113
Word List
114
Word Rerecognizer
115
Ranked List of Words
116
Word Comparer
117
Chosen Word
118
Spelling Hypothesizer
119
Hypothesized Spelling
120
Spelling Merger
121
Spelling List
122
Spelling Rerecognizer
123
Ranked List of Spelling
124
Spelling Evaluator

FIG. 8

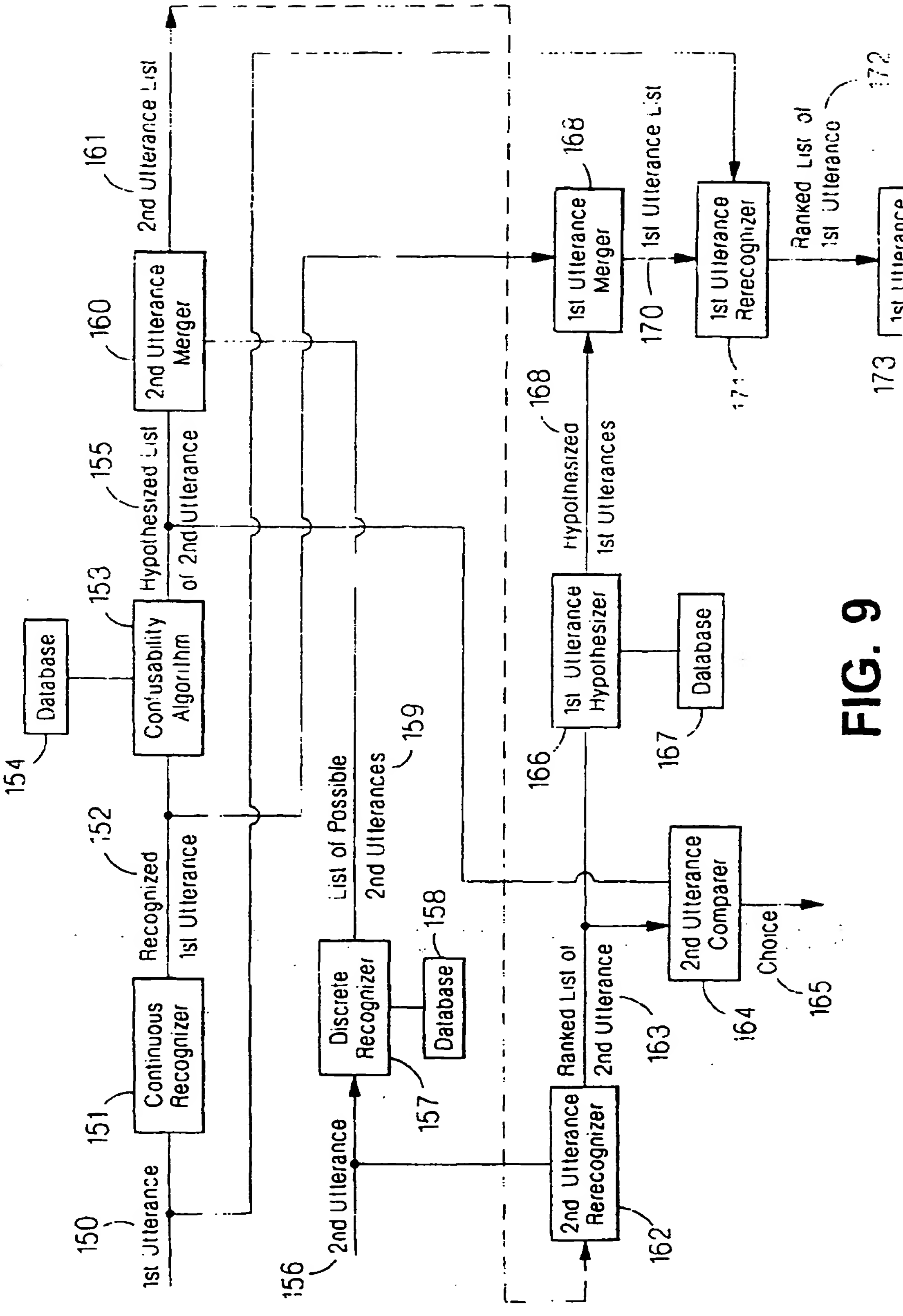

150
1st Utterance
151
Continuous Recognizer
152
Recognized 1st Utterance
153
Confusability Algorithm
154
Database
155
Hypothesized List of 2nd Utterance
160
2nd Utterance Merger
161
2nd Utterance List
156
2nd Utterance
157
Discrete Recognizer
158
Database
159
List of Possible 2nd Utterances
162
2nd Utterance Rerecognizer
163
Ranked List of 2nd Utterance
164
2nd Utterance Comparer
165
Choice
166
1st Utterance Hypothesizer
167
Database
168
Hypothesized 1st Utterances
168
1st Utterance Merger
170
1st Utterance List
171
1st Utterance Rerecognizer
172
Ranked List of 1st Utterance
173
1st Utterance Evaluator

FIG. 9

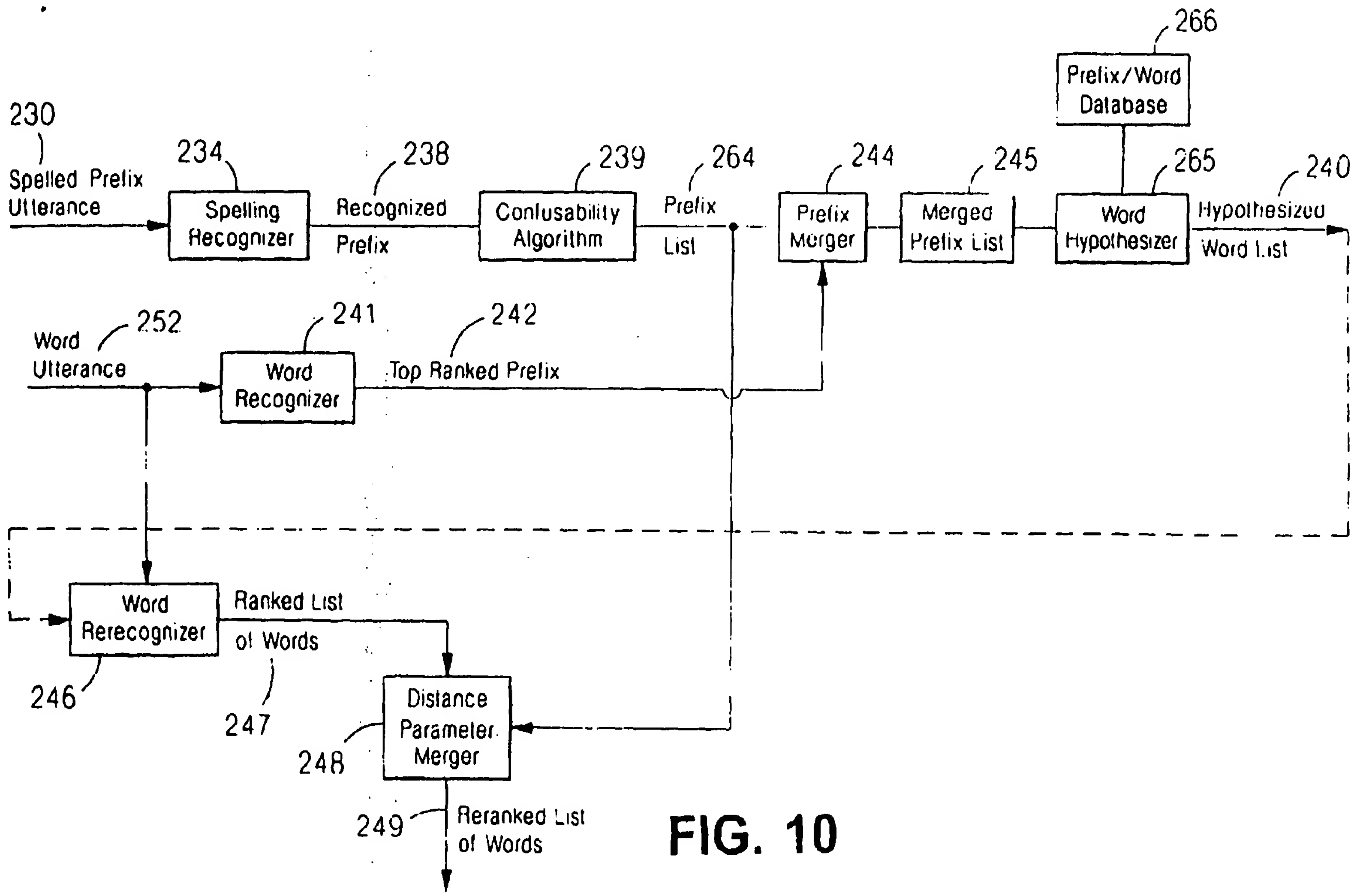

230
Spelled Prefix Utterance
234
Spelling Recognizer
238
Recognized Prefix
239
Confusability Algorithm
264
Prefix List
244
Prefix Merger
245
Merged Prefix List
266
Prefix/Word Database
265
Word Hypothesizer
240
Hypothesized Word List
252
Word Utterance
241
Word Recognizer
242
Top Ranked Prefix
246
Word Rerecognizer
Ranked List of Words
247
248
Distance Parameter Merger
249
Reranked List of Words

FIG. 10

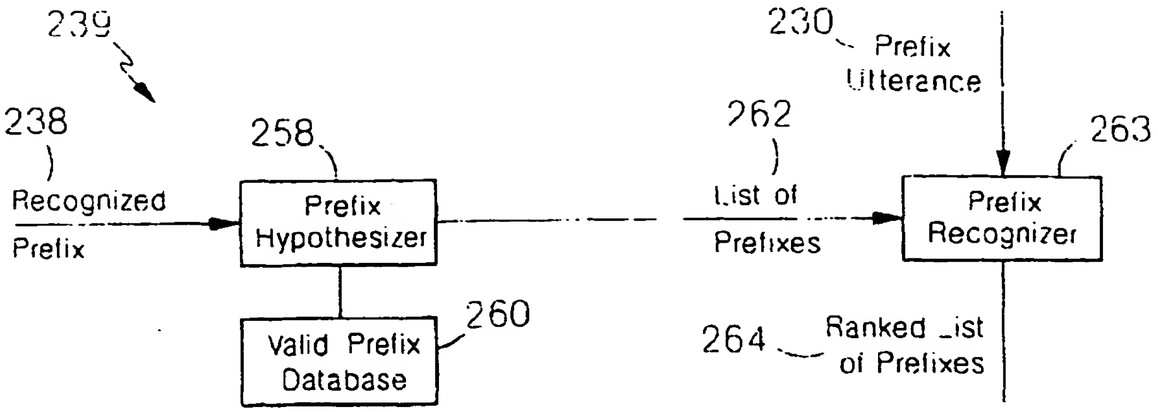

239
238
Recognized
Prefix
258
Prefix
Hypothesizer
Valid Prefix
Database
260
230
Prefix
Utterance
262
List of
Prefixes
263
Prefix
Recognizer
264
Ranked List
of Prefixes

FIG. 11

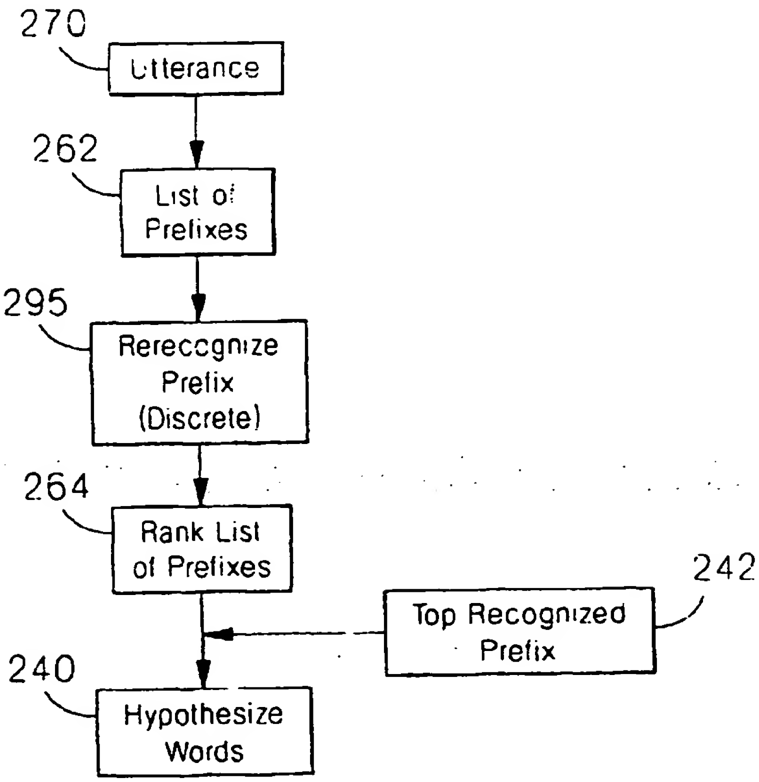

270
Utterance
262
List of
Prefixes
295
Rerecognize
Prefix
(Discrete)
264
Rank List
of Prefixes
242
Top Recognized
Prefix
240
Hypothesize
Words

FIG. 12

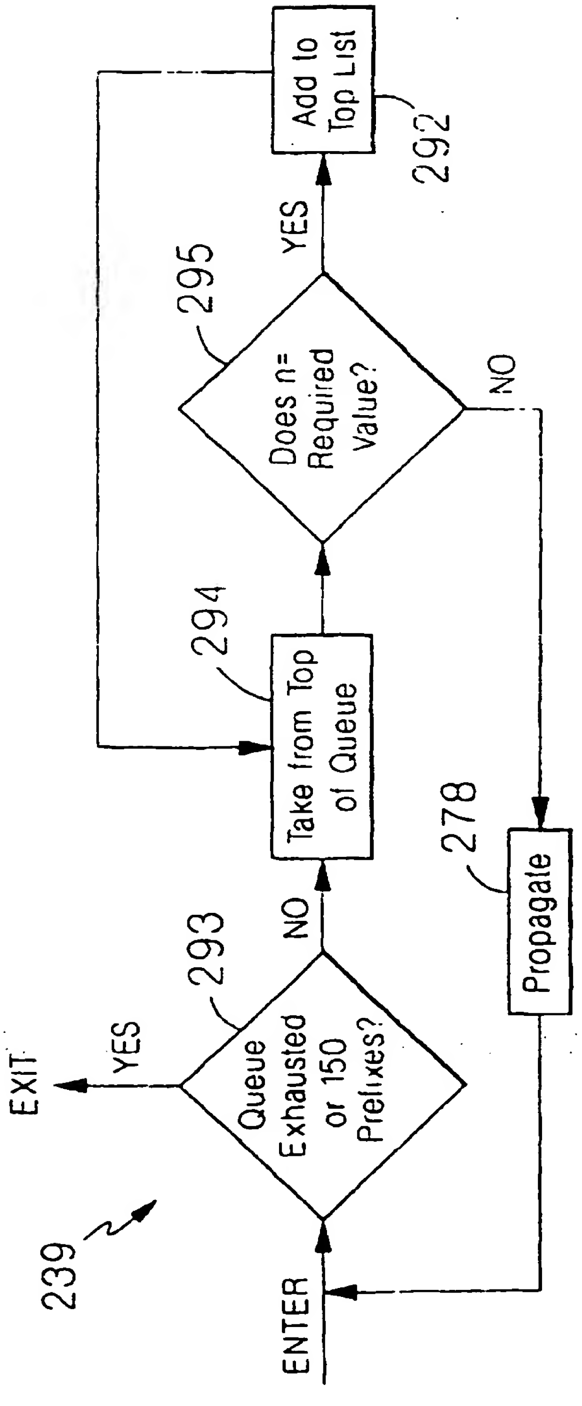

ENTER
239
Queue Exhausted or 150 Prefixes?
293
YES
EXIT
NO
Take from Top of Queue
294
Does n= Required Value?
295
YES
Add to Top List
292
NO
Propagate
278

FIG. 12a

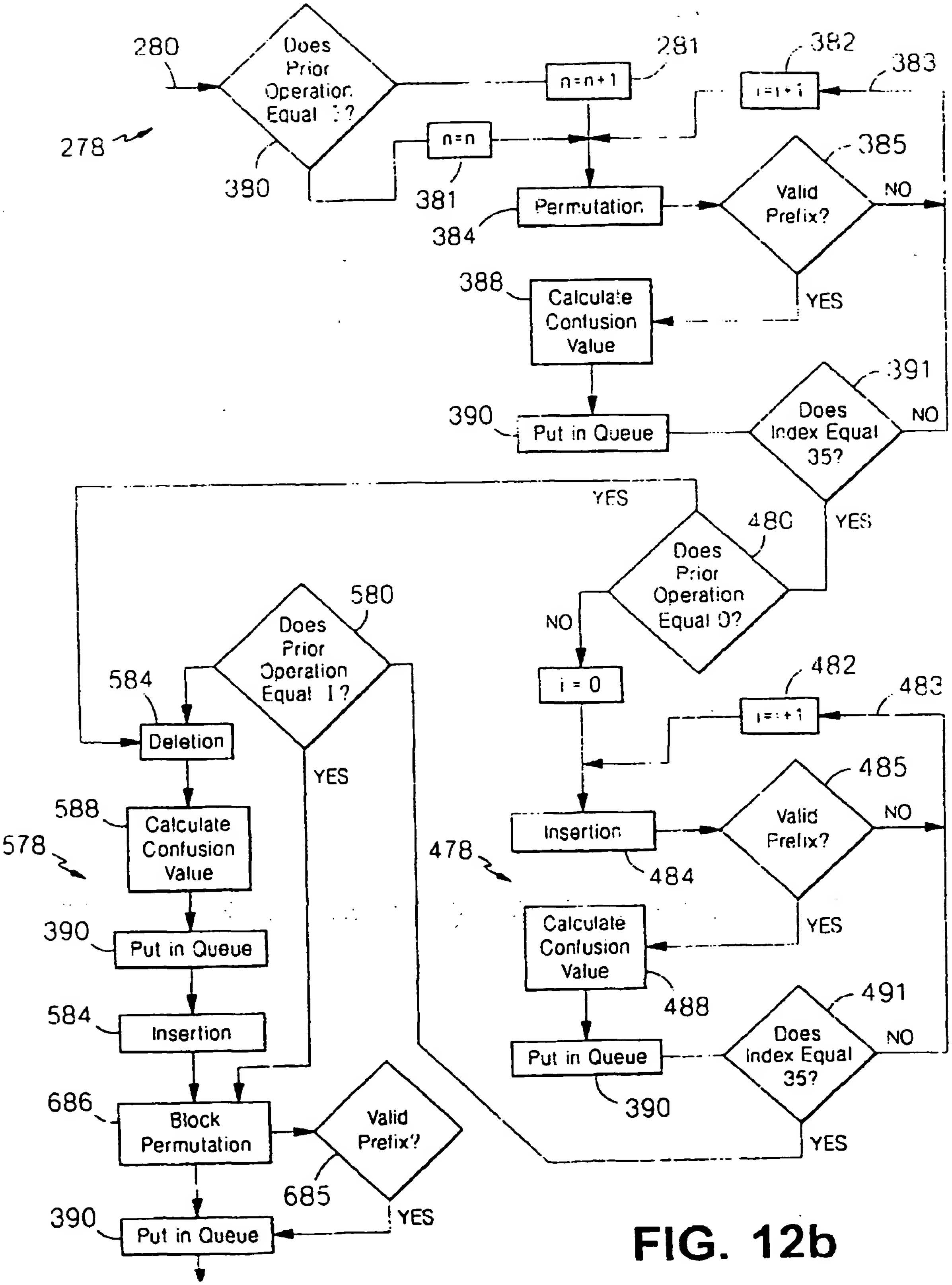

280
Does Prior Operation Equal 1?
278
380
n=n+1
281
n=n
381
i=i+1
382
383
Permutation
384
Valid Prefix?
385
NO
YES
388
Calculate Confusion Value
390
Put in Queue
Does Index Equal 35?
391
NO
YES
YES
480
Does Prior Operation Equal 0?
NO
i = 0
i=i+1
482
483
Insertion
484
Valid Prefix?
485
NO
YES
478
Calculate Confusion Value
488
Put in Queue
390
Does Index Equal 35?
491
NO
YES
580
Does Prior Operation Equal 1?
YES
584
Deletion
588
Calculate Confusion Value
578
390
Put in Queue
584
Insertion
686
Block Permutation
Valid Prefix?
685
YES
390
Put in Queue

FIG. 12b

Permutation

# FIG. 13-1

```
B
     sil    a      b      c      d      e      f      g      h      i      j      k      l      m      n      o      p      q      r
s     1     u      v      w      x      y      z      0      1      2      3      4      5      6      7      8      9
sil 0.000000 0.002951 0.655987 0.000447 0.011988 0.011813 0.000168 0.001145 0.000191 0.003321 0.000025 0.000330
0.002412 0.001016 0.000613 0.004323 0.007841 0.000051 0.002165 0.003379 0.001323 0.001689 0.001680 0.000165 0.000531
0.007866 0.000158 0.000036 0.000018 0.000541 0.000013 0.000013 0.000053 0.000007 0.000038 0.001019 0.000016
 a 0.000000 0.019282 0.530218 0.000610 0.010789 0.017751 0.000224 0.001030 0.000176 0.002747 0.000166 0.001532
0.002171 0.000914 0.000552 0.004990 0.008293 0.000060 0.003522 0.005117 0.002705 0.003631 0.005700 0.000148 0.000644
0.007080 0.000163 0.000032 0.000292 0.000487 0.000012 0.000012 0.000048 0.000006 0.000034 0.025897 0.000054
 b 0.000000 0.003096 0.452850 0.000402 0.009300 0.009422 0.000151 0.001030 0.000172 0.009661 0.000023 0.000297
0.002171 0.000914 0.000552 0.004522 0.007057 0.000046 0.001948 0.003041 0.001526 0.001520 0.019351 0.000148 0.000478
0.037817 0.000142 0.000032 0.000016 0.000487 0.000012 0.000012 0.000048 0.000006 0.000034 0.001101 0.000014
 c 0.000000 0.002656 0.086040 0.002740 0.010789 0.013360 0.000140 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.010860 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000020 0.000010 0.000048 0.000010 0.000110 0.000917 0.000014
 d 0.000000 0.002656 0.036800 0.000350 0.010789 0.010831 0.001410 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.004650 0.000552 0.003891 0.007890 0.000046 0.001948 0.003041 0.001000 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000030 0.000487 0.000010 0.000030 0.000048 0.000010 0.000034 0.000917 0.000014
 e 0.000000 0.011145 0.200916 0.006667 0.017673 0.097556 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.008575 0.000914 0.000552 0.003891 0.048040 0.000046 0.008971 0.011736 0.022734 0.003090 0.001512 0.000875 0.000478
0.009143 0.000142 0.000032 0.000016 0.000487 0.000012 0.000012 0.000048 0.000008 0.000034 0.000917 0.000014
 f 0.000000 0.002656 0.030420 0.000402 0.010789 0.010831 0.000160 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000020
 g 0.000000 0.002656 0.000332 0.000402 0.175007 0.009246 0.000151 0.040180 0.005769 0.025523 0.000023 0.000297
0.002171 0.000914 0.000552 0.010004 0.007057 0.000046 0.001868 0.003041 0.001190 0.001520 0.006230 0.000148 0.000478
0.103141 0.000142 0.000032 0.000016 0.000487 0.000012 0.000012 0.000048 0.000006 0.000034 0.000917 0.000014
 h 0.000000 0.002656 0.000470 0.000402 0.018850 0.012150 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.004800 0.001190 0.001520 0.001512 0.000148 0.000478
0.012100 0.000142 0.000032 0.000016 0.017730 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
 i 0.000000 0.002656 0.506389 0.000402 0.010789 0.030317 0.000151 0.001030 0.000321 0.028698 0.000051 0.000297
0.002350 0.000914 0.000858 0.003891 0.008311 0.000363 0.001560 0.011373 0.001020 0.001520 0.003698 0.000148 0.000398
0.016871 0.000142 0.000032 0.000016 0.000487 0.000012 0.000012 0.001598 0.000008 0.000034 0.000917 0.000014
```

```
j 0.000000 0.002656 0.000180 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
k 0.000000 0.003970 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.006680 0.000046 0.001948 0.003041 0.001190 0.004530 0.001512 0.000148 0.000478
0.007080 0.004160 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
l 0.000000 0.015712 0.399558 0.000667 0.024812 0.040898 0.000151 0.001030 0.000172 0.003750 0.000023 0.000726
0.040527 0.000914 0.000552 0.003891 0.007057 0.000046 0.009596 0.002847 0.001190 0.003716 0.001512 0.000148 0.000478
0.012167 0.000142 0.000032 0.000034 0.000487 0.000012 0.000012 0.000048 0.000008 0.000034 0.001976 0.000014
m 0.000000 0.004779 0.524063 0.000402 0.010789 0.027818 0.003529 0.001030 0.000172 0.002988 0.000022 0.000297
0.002171 0.029505 0.003028 0.019313 0.007057 0.000046 0.001948 0.006416 0.001190 0.002084 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000017 0.000487 0.000012 0.000012 0.000048 0.000006 0.000034 0.000917 0.000014
n 0.000000 0.003610 0.289960 0.003010 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.004100
0.002171 0.001820 0.012780 0.004330 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.005650 0.000142 0.000040 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.001660 0.000014
o 0.000000 0.005003 0.430380 0.000402 0.010045 0.010631 0.000151 0.001030 0.000172 0.008249 0.000208 0.000297
0.003734 0.000914 0.002154 0.057895 0.007057 0.000046 0.002434 0.002981 0.001928 0.001520 0.010751 0.000148 0.000478
0.009843 0.000142 0.000032 0.000016 0.000487 0.000159 0.000012 0.000048 0.000008 0.000034 0.000917 0.000019
p 0.000000 0.002856 0.023110 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.001050 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
q 0.000000 0.002656 0.000230 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
r 0.000000 0.002168 0.465430 0.000402 0.014219 0.024486 0.000151 0.001030 0.000172 0.006015 0.000136 0.000424
0.002171 0.000914 0.000552 0.010530 0.007057 0.000046 0.031591 0.026662 0.001015 0.001520 0.001512 0.000148 0.000478
0.010167 0.000142 0.000032 0.000016 0.000487 0.000052 0.000228 0.000077 0.000008 0.000034 0.000917 0.000140
s 0.000000 0.005949 0.467962 0.000793 0.017136 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.005755 0.000914 0.000552 0.003891 0.027621 0.000046 0.004919 0.031903 0.001190 0.004730 0.001278 0.000148 0.000478
0.023613 0.000142 0.000032 0.000016 0.000750 0.000012 0.000012 0.000048 0.000005 0.000034 0.000917 0.000014
t 0.000000 0.002656 0.020020 0.000402 0.010789 0.010631 0.000220 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000490 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
u 0.000000 0.002656 0.415888 0.000668 0.012103 0.048972 0.000151 0.001030 0.000181 0.005085 0.000023 0.003336
0.012638 0.000914 0.000679 0.003891 0.011001 0.001162 0.004283 0.006677 0.002964 0.034373 0.002477 0.002094 0.000478
0.007080 0.000142 0.000032 0.000024 0.000487 0.000012 0.000012 0.000048 0.000006 0.000034 0.002872 0.000014
v 0.000000 0.002656 0.017460 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
```

FIG. 13-2

```
w 0.000000 0.002656 0.006010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
x 0.000000 0.002656 0.634766 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.029831 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.002716 0.017744
0.007080 0.000142 0.000032 0.000016 0.000487 0.000012 0.000012 0.000048 0.000006 0.000034 0.000917 0.000014
y 0.000000 0.023071 0.207489 0.000402 0.058899 0.062614 0.000151 0.001030 0.000172 0.025780 0.000023 0.000297
0.003352 0.000914 0.001027 0.003891 0.020795 0.000046 0.001948 0.010401 0.011440 0.001520 0.001512 0.000148 0.000478
0.047379 0.000142 0.001046 0.000016 0.000487 0.000012 0.000012 0.000048 0.000006 0.001046 0.001046 0.000014
z 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
0 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
1 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
2 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
3 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
4 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
5 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
6 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
7 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
8 0.000000 0.002656 0.000010 0.000402 0.010789 0.010631 0.000151 0.001030 0.000172 0.002988 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000046 0.001948 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014
```

FIG. 13-3

```
 9 0.000000 0.002856 0.000010 0.000402 0.010789 0.010831 0.000151 0.001030 0.000172 0.002888 0.000023 0.000297
0.002171 0.000914 0.000552 0.003891 0.007057 0.000048 0.001848 0.003041 0.001190 0.001520 0.001512 0.000148 0.000478
0.007080 0.000142 0.000032 0.000016 0.000487 0.000010 0.000010 0.000048 0.000010 0.000034 0.000917 0.000014

O
    sil   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r
s   t   u   v   w   x   y   z   0   1   2   3   4   5   6   7   8   9
sil 0.000000 0.002034 0.001298 0.001324 0.001187 0.004479 0.000810 0.000926 0.000648 0.002472 0.000570 0.001950
0.007608 0.001340 0.004075 0.873477 0.002064 0.000670 0.003242 0.001717 0.002862 0.002068 0.000708 0.001110 0.001384
0.001599 0.000608 0.000015 0.000582 0.000165 0.000061 0.000287 0.000029 0.000004 0.000024 0.002750 0.000017
 a 0.000000 0.018853 0.000837 0.003445 0.001068 0.004031 0.000729 0.000833 0.000583 0.007566 0.000513 0.002074
0.022414 0.001206 0.020971 0.414199 0.001857 0.007815 0.003358 0.002808 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000015 0.016244 0.000148 0.000055 0.000256 0.000026 0.000003 0.000022 0.066354 0.000075
 b 0.000000 0.001831 0.023924 0.001192 0.001068 0.003528 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.002023 0.003668 0.618126 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.016075 0.002409
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000028 0.000003 0.000022 0.002475 0.000015
 c 0.000000 0.004628 0.001168 0.017850 0.001068 0.005087 0.001852 0.000833 0.000583 0.002225 0.000513 0.001755
0.007624 0.001206 0.002925 0.592279 0.002235 0.000803 0.004085 0.003357 0.002576 0.003769 0.000637 0.004208 0.001246
0.001439 0.000546 0.000013 0.000524 0.000242 0.000055 0.000538 0.000026 0.000003 0.000022 0.013548 0.000015
 d 0.000000 0.004287 0.006011 0.001192 0.017020 0.027758 0.000729 0.002550 0.000583 0.002225 0.000513 0.001755
0.006772 0.004913 0.007728 0.504343 0.017450 0.000603 0.005595 0.001545 0.002923 0.003614 0.000637 0.002787 0.002605
0.012972 0.001531 0.000013 0.000524 0.000148 0.000055 0.003088 0.000026 0.000003 0.000022 0.003565 0.000020
 e 0.000000 0.001831 0.001833 0.001192 0.008091 0.036378 0.000728 0.000833 0.000583 0.002225 0.000513 0.001755
0.011255 0.001206 0.014544 0.513147 0.024011 0.000603 0.006442 0.001545 0.010950 0.001861 0.000940 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000003 0.000022 0.002475 0.000015
 f 0.000000 0.002048 0.001168 0.001192 0.001068 0.004031 0.017519 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.647941 0.001857 0.000603 0.009816 0.001545 0.002576 0.002961 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000029 0.000003 0.000022 0.002475 0.000015
 g 0.000000 0.002469 0.001168 0.001192 0.001068 0.004133 0.000729 0.020401 0.000788 0.002271 0.000513 0.001755
0.006848 0.001206 0.008442 0.643866 0.001857 0.000603 0.004228 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000546 0.000148 0.000055 0.000535 0.000028 0.000003 0.000022 0.002475 0.000015
 h 0.000000 0.002102 0.001168 0.001192 0.000862 0.003447 0.000728 0.000833 0.020083 0.002225 0.001383 0.001755
0.006848 0.001206 0.003332 0.628656 0.001857 0.000603 0.002918 0.003813 0.002576 0.001861 0.001335 0.002849 0.001246
0.001439 0.000548 0.000013 0.000524 0.001054 0.000055 0.002950 0.000026 0.000003 0.000022 0.002475 0.000015
 i 0.000000 0.005676 0.001168 0.001192 0.001068 0.010652 0.000728 0.000833 0.000583 0.028732 0.000513 0.001755
0.015640 0.001356 0.008830 0.542111 0.001857 0.000603 0.008270 0.002201 0.004358 0.001670 0.000637 0.000999 0.001246
0.001439 0.000546 0.000058 0.000524 0.000148 0.000055 0.000258 0.000441 0.000003 0.000022 0.002136 0.000103
 j 0.000000 0.001831 0.000971 0.001192 0.001068 0.013320 0.000729 0.000833 0.000583 0.005689 0.018824 0.005758
0.014970 0.001200 0.003314 0.615227 0.001857 0.000603 0.002818 0.001237 0.002576 0.003386 0.000516 0.000999 0.001246
0.002507 0.000546 0.000013 0.000524 0.000541 0.000055 0.000258 0.000026 0.000003 0.000022 0.002475 0.000030
 k 0.000000 0.006666 0.001168 0.001192 0.001068 0.004031 0.000728 0.000833 0.000583 0.021831 0.000513 0.048156
0.081499 0.001208 0.003668 0.372154 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.004744 0.000999 0.001246
0.002981 0.000546 0.000069 0.000524 0.000146 0.000055 0.000258 0.000028 0.000003 0.000022 0.004473 0.000078
 l 0.000000 0.001831 0.001188 0.001593 0.001088 0.004031 0.000729 0.001814 0.000583 0.002225 0.000513 0.001755
0.013970 0.001206 0.003868 0.689783 0.001857 0.000803 0.002822 0.004593 0.002206 0.001861 0.001051 0.000893 0.001246
0.001853 0.000546 0.000013 0.000524 0.000388 0.000055 0.000714 0.000111 0.000003 0.000022 0.002475 0.000015
```

FIG. 13-4

```
m 0.000000 0.001831 0.001168 0.001475 0.001312 0.004031 0.000769 0.000833 0.000583 0.002225 0.000513 0.001973
0.008848 0.014803 0.003668 0.856025 0.001857 0.000603 0.002389 0.002006 0.003660 0.001673 0.002168 0.000999 0.001246
0.001313 0.000546 0.000013 0.000524 0.000148 0.000055 0.000871 0.000026 0.000003 0.000022 0.002475 0.000015
n 0.000000 0.001797 0.001844 0.001192 0.002988 0.003315 0.000729 0.000841 0.000583 0.006032 0.000513 0.001755
0.033921 0.012969 0.026192 0.496047 0.001857 0.000603 0.008565 0.006554 0.002576 0.001827 0.001315 0.004632 0.001246
0.003426 0.000546 0.000011 0.000524 0.000148 0.000073 0.000258 0.000165 0.000003 0.000022 0.002475 0.000026
o 0.000000 0.001831 0.001168 0.000994 0.002128 0.004031 0.001592 0.000833 0.001380 0.002225 0.000513 0.003701
0.006848 0.001128 0.003668 0.577351 0.001593 0.000603 0.008089 0.001919 0.002680 0.001861 0.000570 0.000999 0.001246
0.001439 0.000546 0.000013 0.001162 0.000148 0.000055 0.001075 0.000138 0.000003 0.000022 0.002475 0.000015
p 0.000000 0.002625 0.001168 0.002155 0.000860 0.006693 0.000601 0.000833 0.000583 0.002225 0.000513 0.001913
0.014445 0.001206 0.003668 0.608433 0.014677 0.001016 0.005203 0.003112 0.005428 0.003778 0.000637 0.000999 0.001246
0.001883 0.000546 0.000013 0.000524 0.000475 0.000055 0.000258 0.000027 0.000003 0.000605 0.002475 0.000033
q 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000028 0.000010 0.000022 0.002475 0.000015
r 0.000000 0.002322 0.000987 0.001227 0.001068 0.004031 0.001301 0.000845 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.681570 0.001857 0.000603 0.013857 0.002147 0.002576 0.001861 0.001190 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000003 0.000022 0.002475 0.000019
s 0.000000 0.001751 0.001168 0.001192 0.001068 0.004031 0.003580 0.000715 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003044 0.647349 0.001857 0.000603 0.003777 0.016866 0.002576 0.004769 0.000637 0.000999 0.001584
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000003 0.000022 0.002475 0.000015
t 0.000000 0.001831 0.001168 0.006605 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001690 0.003007 0.596605 0.004424 0.004985 0.005520 0.006306 0.020282 0.002039 0.001678 0.000999 0.001246
0.001439 0.000546 0.000020 0.000767 0.000162 0.000055 0.000258 0.000028 0.000003 0.000022 0.002475 0.000027
u 0.000000 0.001831 0.001168 0.001192 0.001068 0.006034 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.577312 0.002199 0.000519 0.002918 0.001545 0.023369 0.028146 0.000637 0.000999 0.001246
0.001439 0.000546 0.000225 0.000524 0.002723 0.000055 0.000225 0.000028 0.000003 0.000022 0.002475 0.000015
v 0.000000 0.002520 0.004430 0.001528 0.002782 0.007665 0.000729 0.004348 0.000583 0.004455 0.000513 0.001755
0.019663 0.001206 0.003668 0.624418 0.001857 0.000603 0.005720 0.001545 0.005158 0.001679 0.006775 0.000999 0.001246
0.001377 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000013 0.000098 0.002475 0.000015
w 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001075 0.003668 0.728704 0.001857 0.000603 0.006341 0.001545 0.002576 0.001861 0.000637 0.006707 0.001246
0.001439 0.000546 0.000013 0.000460 0.000148 0.000055 0.000258 0.000026 0.000003 0.000022 0.002475 0.000015
x 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000865 0.000833 0.000583 0.003039 0.000513 0.001755
0.006848 0.001318 0.017567 0.582763 0.001857 0.000603 0.002918 0.001298 0.008990 0.001861 0.000637 0.000999 0.027691
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000003 0.000022 0.005218 0.000061
y 0.000000 0.006214 0.001168 0.001192 0.001068 0.005687 0.000729 0.001056 0.000583 0.002225 0.000513 0.001755
0.013837 0.001206 0.003990 0.574684 0.001857 0.009379 0.003385 0.001545 0.002576 0.011212 0.000637 0.000999 0.001246
0.021890 0.000548 0.000013 0.000524 0.000148 0.002014 0.000258 0.000028 0.000003 0.000022 0.002475 0.000015
```

FIG. 13-5

```
 z 0.000000 0.003504 0.001168 0.005415 0.001068 0.011638 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.002543 0.008937 0.617612 0.001857 0.000603 0.003318 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.002061 0.019940 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000003 0.000022 0.002475 0.000015
 0 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
 1 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000728 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001208 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000548 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
 2 0.000000 0.001831 0.001168 0.001192 0.001088 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
 3 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
 4 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
 5 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
 6 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000548 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
 7 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
 8 0.000000 0.001831 0.001168 0.001192 0.001068 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001206 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
 9 0.000000 0.001831 0.001188 0.001192 0.001088 0.004031 0.000729 0.000833 0.000583 0.002225 0.000513 0.001755
0.006848 0.001208 0.003668 0.000010 0.001857 0.000603 0.002918 0.001545 0.002576 0.001861 0.000637 0.000999 0.001246
0.001439 0.000546 0.000013 0.000524 0.000148 0.000055 0.000258 0.000026 0.000010 0.000022 0.002475 0.000015
S
      sil    a     b     c     d     e     f     g     h     i     j     k     l     m     n     o     p     q     r
s     t     u     v     w     x     y     z     0     1     2     3     4     5     6     7     8     9
sil 0.000000 0.004293 0.001955 0.001499 0.001677 0.007384 0.002557 0.001206 0.001291 0.003364 0.000120 0.001343
0.002461 0.001100 0.001950 0.003245 0.001713 0.000984 0.004380 0.889866 0.002084 0.001822 0.000409 0.000860 0.004758
0.001633 0.000618 0.000008 0.000148 0.000373 0.000303 0.000255 0.000033 0.000115 0.000455 0.000869 0.000172
```

FIG. 13-6

```
a 0.000000 0.023289 0.001760 0.001349 0.001510 0.030871 0.002301 0.001085 0.003187 0.003028 0.000305 0.003354
0.002215 0.002389 0.001755 0.002921 0.001891 0.000885 0.003942 0.573972 0.004287 0.001911 0.000368 0.000774 0.004283
0.001470 0.000556 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.004576 0.000155
b 0.000000 0.003864 0.015674 0.001349 0.005781 0.009242 0.002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002530 0.001743 0.001755 0.004628 0.003919 0.000885 0.003942 0.654923 0.001875 0.003154 0.004455 0.000774 0.004283
0.001470 0.000556 0.000007 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.001414 0.000155
c 0.000000 0.003864 0.001760 0.026806 0.003835 0.006655 0.006899 0.001085 0.001162 0.002506 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0.005434 0.000885 0.003942 0.611462 0.001875 0.001640 0.000368 0.000774 0.006459
0.001470 0.000556 0.000007 0.000131 0.000336 0.005702 0.000229 0.000030 0.000104 0.005683 0.000782 0.000155
d 0.000000 0.003864 0.013306 0.001349 0.014492 0.012508 0.002699 0.004916 0.001182 0.003028 0.000108 0.001208
0.002851 0.000990 0.003598 0.002956 0.008057 0.000885 0.004277 0.598122 0.007610 0.002877 0.002257 0.000774 0.005391
0.001470 0.000556 0.000007 0.000203 0.000336 0.000273 0.000229 0.000030 0.000213 0.000409 0.001031 0.000155
e 0.000000 0.003864 0.006517 0.001399 0.007748 0.033189 0.002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002729 0.000990 0.001755 0.002921 0.002925 0.000885 0.003942 0.585748 0.003204 0.001407 0.002677 0.000774 0.004283
0.001470 0.000556 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0.000155
I 0.000000 0.010485 0.001760 0.001349 0.001510 0.006655 0.045208 0.003520 0.003522 0.003203 0.000359 0.001354
0.026531 0.000990 0.001755 0.002489 0.001541 0.000885 0.003942 0.488323 0.001875 0.001640 0.000368 0.000774 0.004283
0.001293 0.000556 0.000007 0.000131 0.000336 0.000273 0.004188 0.000030 0.000104 0.000409 0.000782 0.000155
g 0.000000 0.003864 0.001780 0.001349 0.001510 0.009151 0.002301 0.030765 0.001162 0.003028 0.000108 0.001208
0.002215 0.000990 0.009344 0.002921 0.001673 0.000885 0.003942 0.553366 0.004889 0.001640 0.000368 0.000774 0.030089
0.001470 0.000556 0.000035 0.000131 0.000338 0.000273 0.000229 0.000030 0.000104 0.000524 0.000992 0.000155
h 0.000000 0.003864 0.001760 0.005747 0.001472 0.006655 0.011126 0.002949 0.025652 0.006715 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.542007 0.004799 0.001640 0.000368 0.000774 0.023056
0.001470 0.000556 0.000007 0.000131 0.000336 0.001046 0.000257 0.000030 0.000355 0.001059 0.008846 0.000155
i 0.000000 0.022428 0.001760 0.001286 0.001510 0.009089 0.002301 0.001085 0.002231 0.025378 0.000142 0.001596
0.002215 0.000866 0.001755 0.002921 0.001536 0.000885 0.003942 0.574571 0.001532 0.001640 0.000368 0.000774 0.004283
0.002971 0.000556 0.000016 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.002759 0.000809
j 0.000000 0.003864 0.001760 0.001349 0.001510 0.006855 0.002301 0.001065 0.001162 0.003028 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.000010 0.001875 0.001640 0.000368 0.000774 0.004283
0.001470 0.000556 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0.000155
k 0.000000 0.003864 0.001760 0.003473 0.001510 0.023628 0.002301 0.001085 0.001162 0.003028 0.000198 0.032569
0.002059 0.000990 0.002006 0.003951 0.001541 0.000885 0.003942 0.553382 0.001875 0.001640 0.000368 0.000774 0.019805
0.001470 0.000556 0.000007 0.000131 0.000336 0.000273 0.000229 0.000058 0.000104 0.000409 0.000782 0.000155
I 0.000000 0.003864 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0.002111 0.021339 0.000108 0.001208
0.027585 0.000990 0.001755 0.011560 0.001541 0.000885 0.003942 0.582203 0.001875 0.001640 0.000368 0.001221 0.004283
0.005416 0.000556 0.000007 0.000644 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0.000155
m 0.000000 0.006033 0.001760 0.003976 0.001510 0.008736 0.002301 0.001085 0.001162 0.003987 0.000108 0.001208
0.004585 0.024912 0.013511 0.003071 0.001541 0.000885 0.004842 0.588167 0.001875 0.001640 0.000368 0.000774 0.005753
0.001649 0.000556 0.000007 0.000311 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0.000155
```

FIG. 13-7

```
 n 0.000000 0.011323 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0.001162 0.005636 0.000108 0.001208
0.002215 0.004040 0.027540 0.002895 0.001541 0.000885 0.003942 0.604695 0.001875 0.001640 0.000368 0.000774 0.004283
0.001372 0.000558 0.000007 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.003024 0.000155
 o 0.000000 0.011723 0.001724 0.001348 0.001510 0.006855 0.002301 0.001085 0.000889 0.002789 0.000108 0.001421
0.002320 0.000990 0.001755 0.041892 0.001541 0.000885 0.005702 0.539338 0.001790 0.001640 0.002239 0.000774 0.004283
0.001470 0.000556 0.000008 0.000261 0.000336 0.000273 0.000228 0.000107 0.000104 0.000409 0.000641 0.000155
 p 0.000000 0.003884 0.001760 0.001349 0.004273 0.019405 0.002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002215 0.001981 0.001755 0.002921 0.021866 0.000972 0.003942 0.623632 0.002532 0.001640 0.000368 0.000774 0.006362
0.001470 0.000556 0.000007 0.000131 0.000338 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0.000155
 q 0.000000 0.003864 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.001240 0.001875 0.001640 0.000368 0.000774 0.004283
0.001470 0.000556 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0.000155
 r 0.000000 0.004037 0.001780 0.003502 0.001510 0.007679 0.002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002215 0.000980 0.001755 0.010987 0.001541 0.000885 0.033193 0.573761 0.004203 0.001640 0.000368 0.000774 0.005252
0.001470 0.000558 0.000007 0.000131 0.000338 0.000507 0.000229 0.000041 0.000104 0.000408 0.000782 0.000155
 s 0.000000 0.007070 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0.001162 0.004376 0.000108 0.001208
0.002215 0.000980 0.001755 0.004738 0.001541 0.000885 0.003842 0.616648 0.001875 0.002042 0.000368 0.000774 0.004283
0.001470 0.000556 0.000007 0.000131 0.000336 0.000273 0.000228 0.000030 0.000104 0.000409 0.000721 0.000155
 t 0.000000 0.003864 0.001760 0.002001 0.001510 0.008202 0.002800 0.001085 0.001162 0.003028 0.000108 0.001208
0.001776 0.000990 0.001570 0.002921 0.008210 0.000938 0.003942 0.637364 0.023484 0.001640 0.000368 0.000774 0.003909
0.001470 0.000558 0.000041 0.000131 0.001353 0.000528 0.000229 0.000030 0.000104 0.000518 0.001311 0.000155
 u 0.000000 0.003864 0.002332 0.001349 0.001510 0.010584 0.002301 0.001085 0.003725 0.003028 0.000108 0.001208
0.002215 0.000990 0.002725 0.002921 0.002493 0.002005 0.003942 0.608387 0.001875 0.024595 0.000368 0.000774 0.004283
0.002124 0.000558 0.000007 0.000131 0.000338 0.000273 0.000229 0.000030 0.000120 0.000409 0.001116 0.000155
 v 0.000000 0.003864 0.025698 0.001349 0.014585 0.042202 0.002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.012032 0.585311 0.001555 0.001640 0.000820 0.000774 0.004283
0.010959 0.000803 0.000007 0.000131 0.000336 0.000387 0.000228 0.000030 0.000104 0.000409 0.000782 0.000155
 w 0.000000 0.003864 0.001760 0.001349 0.001747 0.005772 0.002301 0.001085 0.001162 0.009531 0.000108 0.001208
0.002215 0.000990 0.001408 0.004537 0.001541 0.000885 0.029440 0.533464 0.001875 0.003508 0.000368 0.028703 0.020889
0.001642 0.000556 0.000007 0.000131 0.001777 0.000273 0.000229 0.000030 0.000104 0.000409 0.000745 0.000155
 x 0.000000 0.003884 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002215 0.000980 0.001755 0.002921 0.001541 0.000885 0.003942 0.000010 0.001875 0.001640 0.000368 0.000774 0.004283
0.001470 0.000556 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0.000155
 y 0.000000 0.010878 0.001760 0.001349 0.001510 0.008730 0.002301 0.001085 0.001162 0.007650 0.000108 0.001364
0.011049 0.000990 0.001755 0.002706 0.001541 0.000885 0.003942 0.615729 0.001875 0.001640 0.000368 0.000774 0.004283
0.025503 0.000556 0.000007 0.000131 0.000336 0.000273 0.000229 0.000027 0.000131 0.000409 0.000782 0.000155
 z 0.000000 0.003884 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.005830 0.001875 0.001640 0.000368 0.000774 0.004283
0.001470 0.000558 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0.000155
```

FIG. 13-8

```
 0 0.000000 0.003864 0.001760 0.001349 0.001288 0.006821 0.002301 0.001085 0.001162 0.014456 0.002963 0.001208
0.002215 0.000890 0.001528 0.007452 0.001541 0.029888 0.043803 0.415858 0.001580 0.018124 0.000368 0.000774 0 021006
0.001470 0.020579 0.000007 0.000217 0.009052 0.001532 0.004475 0.000407 0.002794 0.002651 0.002258 0.005160
 1 0.000000 0.029530 0.001760 0.001349 0.001510 0.006855 0.002301 0.001085 0.001162 0.003028 0.000108 0.002480
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.189230 0.008170 0.001640 0 000368 0 000774 0 004283
0.001470 0.000556 0.000010 0.003130 0.000336 0.000273 0.000228 0.000050 0.000160 0.000409 0.000782 0.000155
 2 0.000000 0.003684 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.000010 0.001875 0.001640 0.000368 0.000774 0 004283
0.001470 0.000558 0.000010 0.000131 0.000338 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0 000155
 3 0.000000 0.003864 0.001760 0.001349 0.001510 0.006855 0.002301 0.001085 0.001162 0.003028 0.000108 0 001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.000010 0.001875 0.001640 0.000368 0 000774 0 004283
0.001470 0.000558 0.000010 0.000131 0.000336 0.000273 0.000228 0.000030 0.000104 0.000409 0.000782 0.000155
 4 0.000000 0.003864 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0 001162 0 003028 0.000108 0 001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.000010 0.001875 0.001640 0.000368 0 000774 0.004283
0.001470 0.000556 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0 000104 0.000409 0.000782 0 000155
 5 0.000000 0.003684 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0 001162 0.003028 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0 000010 0.001875 0.001640 0.000368 0 000774 0 004283
0.001470 0.000558 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0.000104 0.000409 0.000782 0 000155
 6 0.000000 0.003864 0:001760 0.001349 0.001510 0.006655 0.002301 0.001085 0 001162 0.003028 0.000108 0 001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.000010 0.001875 0.001640 0.000368 0.000774 0 004283
0.001470 0.000556 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0 000104 0.000409 0.000782 0 000155
 7 0.000000 0.003864 0.001760 0.001349 0.001510 0.006655 0.002301 0.001085 0 001162 0.003028 0.000108 0 001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.102760 0.001875 0.001640 0.000368 0 000774 0 006170
0.001470 0.000558 0.000010 0.000131 0.000336 0.000273 0.000229 0.000300 0.000300 0.004540 0.000782 0.000155
 8 0.000000 0.003864 0.001760 0.001349 0.001510 0.006655 0 002301 0.001085 0.001162 0.003028 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0.001541 0.000885 0.003942 0.000010 0.001875 0.001640 0.000368 0 000774 0 004283
0.001470 0.000558 0.000010 0.000131 0.000336 0.000273 0.000228 0.000030 0.000104 0.000409 0.000782 0.000155
 9 0.000000 0.003884 0.001780 0.001349 0.001510 0.006655 0.002301 0.001085 0 001162 0.003028 0.000108 0.001208
0.002215 0.000990 0.001755 0.002921 0 001541 0.000885 0 003942 0.000010 0.001875 0.001640 0 000368 0.000774 0 004283
0.001470 0.000558 0.000010 0.000131 0.000336 0.000273 0.000229 0.000030 0 000104 0.000409 0 000782 0.000155
```

FIG. 13-9

B

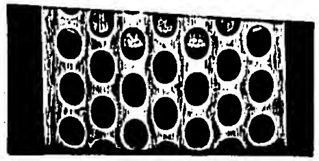
Insertion

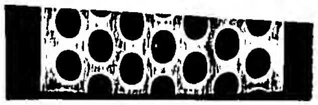
# FIG. 13a-1

```
s        t        u        v        w        x        y        z        0        1        2        3        4        5        6        7        8        9
sil 0.000000 0.003038 0.000000 0.000501 0.007833 0.010596 0.000140 0.002850 0.000154 0.003895 0.000016 0.000186
0.004075 0.002665 0.001353 0.006677 0.003305 0.000043 0.003513 0.004056 0.000984 0.003816 0.001134 0.000148 0.001717
0.006671 0.000230 0.000011 0.000016 0.000388 0.000009 0.000013 0.000042 0.000007 0.000013 0.001281 0.000010
 a 0.000000 0.059830 0.059193 0.000529 0.007050 0.008129 0.000275 0.002385 0.000169 0.003326 0.000135 0.001385
0.003888 0.002398 0.001217 0.006009 0.006808 0.000058 0.002871 0.003850 0.001898 0.003158 0.003214 0.000133 0.001545
0.006004 0.000207 0.000010 0.000271 0.000348 0.000008 0.000011 0.000038 0.000008 0.000012 0.040848 0.000014
 b 0.000000 0.002365 0.113161 0.000451 0.009144 0.008218 0.000128 0.002385 0.000139 0.003326 0.000015 0.000168
0.003868 0.002398 0.001217 0.006009 0.004744 0.000038 0.003162 0.003850 0.001337 0.003434 0.016841 0.000131 0.001545
0.015058 0.000207 0.000010 0.000014 0.000348 0.000008 0.000011 0.000038 0.000006 0.000012 0.001153 0.000009
 c 0.000000 0.002732 0.010180 0.011230 0.007050 0.010830 0.000120 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003850 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000310 0.000010 0.000010 0.000038 0.000010 0.000090 0.001153 0.000010
 d 0.000000 0.002732 0.004410 0.000410 0.008520 0.009538 0.000390 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.007200 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000020 0.000348 0.000010 0.000020 0.000038 0.000010 0.000010 0.001153 0.000010
 e 0.000000 0.005861 0.011675 0.001611 0.014182 0.217112 0.000128 0.002385 0.000224 0.003326 0.000015 0.000168
0.004846 0.002398 0.001217 0.006009 0.036343 0.000038 0.004114 0.003650 0.018599 0.003434 0.001021 0.001503 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000008 0.000011 0.000038 0.000008 0.000012 0.001153 0.000009
 f 0.000000 0.002732 0.003510 0.000451 0.007050 0.009538 0.000760 0.002385 0.000139 0.003326 0.000015 0.000140
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 g 0.000000 0.002732 0.000038 0.000451 0.152543 0.009538 0.000128 0.098715 0.004326 0.019122 0.000015 0.000168
0.003668 0.002398 0.001217 0.014113 0.002975 0.000038 0.003234 0.003850 0.000885 0.003434 0.003805 0.000133 0.001545
0.033448 0.000207 0.000010 0.000014 0.000348 0.000008 0.000011 0.000038 0.000008 0.000012 0.001153 0.000009
 h 0.000000 0.002732 0.000080 0.000451 0.010770 0.010240 0.000128 0.002385 0.000139 0.003326 0.000015 0.000168
0.003868 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.005570 0.000885 0.003434 0.001021 0.000133 0.001545
0.006550 0.000207 0.000010 0.000014 0.012360 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 i 0.000000 0.002732 0.058738 0.000451 0.007050 0.013874 0.000128 0.002385 0.000298 0.099874 0.000060 0.000168
0.003668 0.002398 0.001217 0.006009 0.004835 0.000295 0.003162 0.007827 0.000885 0.003434 0.002715 0.000133 0.001545
0.007930 0.000207 0.000010 0.000014 0.000348 0.000008 0.000011 0.001275 0.000006 0.000012 0.001153 0.000009
 j 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000126 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003850 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 k 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.003940 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.007280 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 l 0.000000 0.004447 0.045011 0.000820 0.021867 0.019677 0.000128 0.002385 0.000139 0.003847 0.000015 0.000891
0.124785 0.002398 0.001217 0.006009 0.003007 0.000038 0.007527 0.003850 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000008 0.000011 0.000038 0.000008 0.000012 0.001805 0.000009
 m 0.000000 0.005456 0.054138 0.000451 0.007050 0.022642 0.003189 0.002385 0.000139 0.003328 0.000020 0.000168
0.003668 0.084812 0.002572 0.007017 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000116 0.001545
0.006004 0.000207 0.000010 0.000032 0.000348 0.000008 0.000011 0.000038 0.000008 0.000012 0.001153 0.000009
```

```
0.004355 0.000038 0.003162 0.003650 0.001198 0.003434 0.008482 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000094 0.000011 0.000038 0.000006 0.000012 0.001153 0.000009
 p 0.000000 0.002732 0.003660 0.000451 0.007050 0.009536 0.000126 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000730 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 q 0.000000 0.002732 0.000030 0.000451 0.007050 0.009536 0.000126 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 r 0.000000 0.002732 0.051948 0.000451 0.012453 0.019381 0.000126 0.002385 0.000121 0.003326 0.000112 0.000194
0.003668 0.002398 0.001217 0.007688 0.002975 0.000038 0.100020 0.007317 0.000885 0.003434 0.001021 0.000133 0.001545
0.013972 0.000207 0.000010 0.000014 0.000348 0.000010 0.000206 0.000038 0.000006 0.000012 0.001153 0.000101
 s 0.000000 0.004833 0.054168 0.000669 0.007050 0.009536 0.000126 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.022243 0.000038 0.003162 0.108541 0.000885 0.004822 0.001021 0.000133 0.001545
0.006095 0.000207 0.000010 0.000014 0.000658 0.000008 0.000011 0.000038 0.000005 0.000012 0.001153 0.000009
 t 0.000000 0.002732 0.003210 0.000451 0.007050 0.009536 0.000126 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.001280 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000440 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 u 0.000000 0.002732 0.048518 0.000592 0.010198 0.017854 0.000126 0.002385 0.000245 0.003326 0.000015 0.000807
0.009143 0.002398 0.001217 0.006009 0.009485 0.000951 0.003162 0.003650 0.001962 0.125245 0.002201 0.002702 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000008 0.000011 0.000038 0.000006 0.000012 0.000923 0.000009
 v 0.000000 0.002732 0.002320 0.000451 0.007050 0.009536 0.000126 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.003060 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 w 0.000000 0.002732 0.000640 0.000451 0.007050 0.009536 0.000126 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 x 0.000000 0.002732 0.070471 0.000451 0.007050 0.009536 0.000126 0.002385 0.000139 0.003326 0.000015 0.000168
0.003139 0.002398 0.001217 0.019874 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000766 0.062837
0.006004 0.000207 0.000010 0.000014 0.000348 0.000008 0.000011 0.000038 0.000006 0.000012 0.001153 0.000009
 y 0.000000 0.017825 0.023452 0.000451 0.033779 0.021472 0.000126 0.002385 0.000139 0.002903 0.000015 0.000168
0.003668 0.002398 0.001814 0.006009 0.015807 0.000038 0.003162 0.007904 0.008070 0.003434 0.001021 0.000133 0.001545
0.153084 0.000207 0.000166 0.000014 0.000348 0.000008 0.000011 0.000038 0.000006 0.000166 0.001153 0.000009
 z 0.000000 0.002732 0.000010 0.000451 0.007050 0.009536 0.000126 0.002385 0.000139 0.003326 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
```

FIG. 13a-2





```
 0 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 1 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 2 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 3 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 4 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 5 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 6 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 7 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 8 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010
 9 0.000000 0.002732 0.000010 0.000451 0.007050 0.009538 0.000128 0.002385 0.000139 0.003328 0.000015 0.000168
0.003668 0.002398 0.001217 0.006009 0.002975 0.000038 0.003162 0.003650 0.000885 0.003434 0.001021 0.000133 0.001545
0.006004 0.000207 0.000010 0.000014 0.000348 0.000010 0.000010 0.000038 0.000010 0.000010 0.001153 0.000010

O      sil    a     b     c     d     e     f     g     h     i     j     k     l     m     n     o     p     q     r
 s     t     u     v     w     x     y     z     0     1     2     3     4     5     6     7     8     9
sil 0.000000 0.002869 0.002764 0.002595 0.002029 0.005812 0.002500 0.002156 0.002162 0.004161 0.001947 0.004285
0.008029 0.002432 0.003572 0.000000 0.002883 0.000403 0.003014 0.002883 0.003135 0.003714 0.001128 0.001080 0.002918
0.002661 0.002208 0.000010 0.000188 0.000219 0.000018 0.000181 0.000020 0.000003 0.000008 0.002464 0.000015
 a 0.000000 0.074705 0.002488 0.008566 0.001828 0.005230 0.002250 0.001941 0.001946 0.007224 0.001753 0.003857
0.017243 0.002189 0.010412 0.021573 0.002595 0.002751 0.008498 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001887 0.000009 0.004028 0.000197 0.000014 0.000183 0.000018 0.000003 0.000007 0.057733 0.000013
 b 0.000000 0.002582 0.083845 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.076405 0.002595 0.000383 0.002712 0.002595 0.002822 0.003342 0.001015 0.005128 0.002628
0.002395 0.001887 0.000009 0.000169 0.000197 0.000014 0.000183 0.000018 0.000003 0.000007 0.002218 0.000013
 c 0.000000 0.004269 0.002488 0.075351 0.001828 0.004970 0.002382 0.001941 0.001946 0.003745 0.001753 0.003857
0.006380 0.002189 0.003215 0.056353 0.002595 0.000363 0.002712 0.004415 0.002822 0.003342 0.001015 0.000786 0.002828
0.002395 0.001887 0.000009 0.000239 0.000197 0.000014 0.000446 0.000018 0.000003 0.000007 0.012062 0.000013
```

FIG. 13a-3

```
d 0.000000 0.002826 0.007048 0.002335 0.060609 0.021489 0.002250 0.002207 0.001946 0.003745 0.001753 0.003857
0.007227 0.002054 0.005242 0.051673 0.016058 0.000355 0.002342 0.002595 0.002298 0.002785 0.001015 0.000945 0.002626
0.004247 0.001987 0.000009 0.000169 0.000197 0.000014 0.001049 0.000018 0.000003 0.000007 0.005543 0.000015
e 0.000000 0.002582 0.002488 0.002335 0.006496 0.127255 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007015 0.002189 0.006537 0.054843 0.018788 0.000363 0.005336 0.002595 0.009289 0.003342 0.001204 0.000972 0.002626
0.002395 0.001987 0.000009 0.000169 0.000197 0.000014 0.000163 0.000018 0.000003 0.000007 0.002218 0.000013
f 0.000000 0.002582 0.002488 0.002335 0.001826 0.005230 0.078570 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.075843 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000009 0.000169 0.000197 0.000014 0.000163 0.000018 0.000003 0.000007 0.002218 0.000013
g 0.000000 0.002582 0.002488 0.002335 0.001826 0.005230 0.002250 0.075069 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.006172 0.086754 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000009 0.000206 0.000197 0.000014 0.000478 0.000018 0.000003 0.000007 0.002218 0.000013
h 0.000000 0.002582 0.002488 0.002335 0.001826 0.005230 0.002250 0.001941 0.077239 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.073777 0.002595 0.000363 0.002712 0.003924 0.002822 0.003342 0.001015 0.000954 0.002626
0.002395 0.001987 0.000009 0.000169 0.001008 0.000014 0.002449 0.000018 0.000003 0.000007 0.002218 0.000013
i 0.000000 0.004348 0.002488 0.002335 0.001826 0.009981 0.002250 0.001941 0.001946 0.107926 0.001753 0.003857
0.011587 0.002189 0.003215 0.053302 0.002595 0.000363 0.006051 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000037 0.000169 0.000197 0.000014 0.000163 0.000324 0.000003 0.000007 0.001809 0.000105
j 0.000000 0.002582 0.002488 0.002335 0.001826 0.008818 0.002250 0.001941 0.001946 0.003745 0.071098 0.003857
0.023270 0.002189 0.003843 0.078395 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000009 0.000169 0.001002 0.000014 0.000163 0.000018 0.000003 0.000007 0.002218 0.000013
k 0.000000 0.002582 0.002488 0.002335 0.001826 0.005230 0.002250 0.001941 0.001946 0.023791 0.001753 0.147025
0.065538 0.002189 0.003215 0.039628 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.007456 0.000972 0.002626
0.002451 0.001987 0.000038 0.000169 0.000197 0.000014 0.000163 0.000018 0.000003 0.000007 0.004880 0.000110
l 0.000000 0.002582 0.002488 0.002335 0.001826 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.052357 0.002189 0.003215 0.094090 0.002595 0.000363 0.002712 0.003585 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000009 0.000296 0.000197 0.000014 0.000428 0.000024 0.000003 0.000007 0.002218 0.000013
m 0.000000 0.002582 0.002488 0.002335 0.001826 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.061876 0.003215 0.091861 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000009 0.000169 0.000197 0.000014 0.000423 0.000018 0.000003 0.000007 0.002218 0.000013
n 0.000000 0.002582 0.002488 0.002335 0.001826 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.037275 0.018888 0.078086 0.054330 0.002595 0.000363 0.002895 0.007172 0.002822 0.003342 0.001015 0.000953 0.002626
0.002395 0.001987 0.000011 0.000182 0.000197 0.000038 0.000163 0.000128 0.000003 0.000007 0.002218 0.000023
o 0.000000 0.002582 0.002488 0.002335 0.001826 0.005230 0.002157 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.173974 0.002595 0.000363 0.008598 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000009 0.000561 0.000197 0.000014 0.000869 0.000123 0.000003 0.000007 0.002218 0.000013
p 0.000000 0.002124 0.002488 0.002335 0.001826 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.016433 0.002189 0.003215 0.075385 0.060877 0.000859 0.002712 0.002595 0.005572 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000009 0.000169 0.000409 0.000014 0.000163 0.000018 0.000003 0.000135 0.002218 0.000031
```

FIG. 13a-4

```
 q 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
 r 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002197 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.100030 0.002595 0.000363 0.058697 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000009 0.000200 0.000197 0.000014 0.000163 0.000018 0.000003 0.000007 0.002218 0.000013
 s 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.004162 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.089847 0.002595 0.000363 0.002712 0.069028 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000009 0.000169 0.000197 0.000014 0.000163 0.000018 0.000003 0.000007 0.002218 0.000013
 t 0.000000 0.002582 0.002488 0.003737 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.067026 0.002782 0.008443 0.002712 0.007697 0.076492 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000008 0.000300 0.000183 0.000014 0.000163 0.000018 0.000003 0.000007 0.002218 0.000015
 u 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.063587 0.002595 0.000709 0.002712 0.002595 0.006657 0.121734 0.001015 0.000972 0.002626
0.002395 0.001987 0.000147 0.000147 0.004914 0.000014 0.000147 0.000018 0.000003 0.000007 0.002218 0.000013
 v 0.000000 0.003290 0.006485 0.002335 0.002435 0.006542 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.019961 0.002189 0.003215 0.096380 0.002595 0.000363 0.006608 0.002595 0.002822 0.003342 0.029164 0.000972 0.002626
0.002395 0.001987 0.000009 0.000169 0.000197 0.000014 0.000163 0.000018 0.000003 0.000015 0.002218 0.000013
 w 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.109616 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.029694 0.002626
0.002395 0.001987 0.000009 0.000378 0.000197 0.000014 0.000163 0.000018 0.000003 0.000007 0.002218 0.000013
 x 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.004713 0.064201 0.002595 0.000363 0.002712 0.002595 0.004575 0.003342 0.001015 0.000972 0.102163
0.002395 0.001987 0.000009 0.000169 0.000197 0.000014 0.000163 0.000018 0.000003 0.000007 0.004683 0.000112
 y 0.000000 0.002871 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.009525 0.002189 0.003215 0.080981 0.002595 0.001555 0.002712 0.002595 0.002822 0.005453 0.001015 0.000972 0.002626
0.083135 0.001987 0.000009 0.000169 0.000197 0.000421 0.000163 0.000018 0.000003 0.000007 0.002218 0.000013
 z 0.000000 0.002582 0.002488 0.005818 0.001828 0.009355 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.004540 0.080880 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.079735 0.000009 0.000169 0.000197 0.000014 0.000163 0.000018 0.000003 0.000007 0.002218 0.000013
 0 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
 1 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
 2 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
```

FIG. 13a-5

```
 3 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
 4 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
 5 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
 6 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
 7 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
 8 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013
 9 0.000000 0.002582 0.002488 0.002335 0.001828 0.005230 0.002250 0.001941 0.001946 0.003745 0.001753 0.003857
0.007227 0.002189 0.003215 0.000010 0.002595 0.000363 0.002712 0.002595 0.002822 0.003342 0.001015 0.000972 0.002626
0.002395 0.001987 0.000010 0.000169 0.000197 0.000014 0.000163 0.000018 0.000010 0.000010 0.002218 0.000013

S     sil   a    b    c    d    e    f    g    h    i    j    k    l    m    n    o    p    q    r
   s    t    u    v    w    x    y    z    0    1    2    3    4    5    6    7    8    9
sil 0.000000 0.006029 0.002795 0.003175 0.002518 0.008698 0.005247 0.003046 0.002695 0.004727 0.000204 0.003352
0.004413 0.002729 0.003775 0.005913 0.003003 0.000805 0.006048 0.000000 0.003110 0.003297 0.000422 0.002795 0.005113
0.003289 0.000532 0.000007 0.000347 0.000310 0.000277 0.000213 0.000028 0.000108 0.000738 0.000749 0.000157
 a 0.000000 0.085120 0.002516 0.002857 0.002264 0.025738 0.004722 0.002741 0.002428 0.004254 0.000159 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.056565 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000006 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.003474 0.000141
 b 0.000000 0.005428 0.058351 0.002857 0.005043 0.008266 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.003307 0.000725 0.005443 0.084913 0.002799 0.002862 0.003853 0.002515 0.004602
0.002960 0.000479 0.000006 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.001304 0.000141
 c 0.000000 0.005426 0.002516 0.098547 0.002888 0.007827 0.006706 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.005372 0.000725 0.005443 0.071393 0.002799 0.002968 0.000380 0.002515 0.006283
0.002960 0.000479 0.000006 0.000312 0.000279 0.005554 0.000191 0.000025 0.000097 0.005540 0.000674 0.000141
 d 0.000000 0.005428 0.010746 0.002857 0.053319 0.012165 0.004740 0.004220 0.002428 0.004254 0.000183 0.003017
0.003972 0.002456 0.002908 0.005322 0.006345 0.000725 0.005443 0.078857 0.006233 0.002484 0.001951 0.002515 0.007847
0.002960 0.000479 0.000008 0.000312 0.000279 0.000249 0.000191 0.000025 0.000298 0.000684 0.000983 0.000141
```

FIG. 13a-6

```
e 0.000000 0.005426 0.005288 0.002857 0.006898 0.118273 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.067989 0.002799 0.002968 0.002268 0.002515 0.004602
0.002960 0.000479 0.000008 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
f 0.000000 0.009870 0.002516 0.002857 0.002264 0.007827 0.147465 0.003448 0.003455 0.004254 0.000283 0.003017
0.031731 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.060360 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000006 0.000312 0.000424 0.000248 0.002755 0.000025 0.000097 0.000664 0.000674 0.000141
g 0.000000 0.005428 0.002516 0.002857 0.002264 0.007723 0.004722 0.101035 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.007733 0.005322 0.002703 0.000725 0.005443 0.071077 0.004616 0.002968 0.000380 0.002515 0.032483
0.002960 0.000479 0.000031 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000820 0.000141
h 0.000000 0.005428 0.002516 0.004806 0.002264 0.007827 0.018548 0.002305 0.089693 0.004254 0.000183 0.003017
0.003972 0.002458 0.003397 0.005322 0.002703 0.000725 0.005443 0.075328 0.003735 0.002968 0.000380 0.002515 0.022223
0.002960 0.000479 0.000006 0.000312 0.000279 0.001020 0.000219 0.000025 0.000383 0.001033 0.009554 0.000141
i 0.000000 0.018361 0.002516 0.002857 0.002264 0.008470 0.004722 0.002741 0.002426 0.099236 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.052080 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000248 0.000191 0.000025 0.000097 0.000664 0.002260 0.001339
j 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.000010 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000478 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
k 0.000000 0.005426 0.002516 0.003044 0.002264 0.018565 0.004722 0.002741 0.002426 0.004254 0.000177 0.112720
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.070194 0.002799 0.002968 0.000380 0.002515 0.020309
0.002960 0.000479 0.000006 0.000312 0.000279 0.000249 0.000191 0.000051 0.000097 0.000664 0.000674 0.000141
l 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.017068 0.000183 0.003017
0.104000 0.002456 0.003397 0.009804 0.002703 0.000725 0.005443 0.087592 0.002799 0.002968 0.000380 0.002515 0.004602
0.003943 0.000479 0.000006 0.000574 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
m 0.000000 0.005167 0.002516 0.002319 0.002264 0.007448 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003473 0.088848 0.011733 0.005322 0.002703 0.000725 0.005443 0.079807 0.002799 0.002968 0.000380 0.002515 0.006943
0.002960 0.000479 0.000008 0.000282 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
n 0.000000 0.010056 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002428 0.004680 0.000183 0.003017
0.003972 0.003297 0.102443 0.005322 0.002703 0.000725 0.005443 0.076185 0.002799 0.002968 0.000380 0.002515 0.004189
0.002960 0.000479 0.000006 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.002750 0.000141
o 0.000000 0.009989 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002428 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.160171 0.002703 0.000725 0.006187 0.042027 0.002799 0.002968 0.001917 0.002515 0.004602
0.002960 0.000479 0.000006 0.000343 0.000279 0.000249 0.000181 0.000069 0.000097 0.000664 0.000674 0.000141
p 0.000000 0.005426 0.002516 0.002857 0.003780 0.017728 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.078874 0.000858 0.005443 0.078238 0.002611 0.002968 0.000380 0.002515 0.007483
0.002960 0.000479 0.000006 0.000312 0.000278 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
q 0.000000 0.005428 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002428 0.004254 0.000183 0.003017
0.003972 0.002458 0.003397 0.005322 0.002703 0.000725 0.005443 0.000230 0.002789 0.002968 0.000380 0.002515 0.004802
0.002960 0.000479 0.000010 0.000312 0.000279 0.000248 0.000191 0.000025 0.000097 0.000664 0.000874 0.000141
```

FIG. 13a-7

```
 r 0.000000 0.005426 0.002516 0.002354 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.007331 0.002703 0.000725 0.116471 0.066302 0.002799 0.002968 0.000380 0.002515 0.003730
0.002960 0.000479 0.000006 0.000312 0.000279 0.000485 0.000191 0.000033 0.000097 0.000664 0.000674 0.000141
 s 0.000000 0.004561 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.004279 0.002703 0.000725 0.005443 0.170850 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000006 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000152
 t 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.006851 0.000826 0.005443 0.078820 0.084105 0.002968 0.000380 0.002515 0.003949
0.002960 0.000479 0.000039 0.000312 0.001045 0.000505 0.000191 0.000025 0.000097 0.000664 0.001123 0.000141
 u 0.000000 0.005426 0.002516 0.002857 0.002264 0.007205 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.003407 0.005443 0.068743 0.002799 0.088425 0.000380 0.002515 0.004602
0.002960 0.000479 0.000006 0.000312 0.000279 0.000249 0.000191 0.000025 0.000208 0.000664 0.000674 0.000141
 v 0.000000 0.005426 0.021467 0.002857 0.012109 0.041350 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.010832 0.096582 0.002799 0.002968 0.003530 0.002515 0.004602
0.009312 0.000688 0.000006 0.000312 0.000279 0.000323 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
 w 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.008404 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.024589 0.053024 0.002799 0.002912 0.000380 0.100518 0.021817
0.002960 0.000479 0.000006 0.000312 0.001603 0.000248 0.000191 0.000025 0.000097 0.000664 0.000549 0.000141
 x 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.000010 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
 y 0.000000 0.009891 0.002516 0.002857 0.002264 0.008719 0.004722 0.002741 0.002426 0.007105 0.000183 0.003017
0.007294 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.072008 0.002799 0.002968 0.000380 0.002515 0.004602
0.096304 0.000479 0.000006 0.000312 0.000288 0.000249 0.000191 0.000025 0.000128 0.000664 0.000674 0.000141
 z 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.000500 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
 0 0.000000 0.005426 0.002516 0.002857 0.002264 0.006238 0.004722 0.002741 0.002426 0.014144 0.008645 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.023420 0.033631 0.057410 0.002799 0.013188 0.000502 0.002515 0.028274
0.002960 0.017989 0.000007 0.000353 0.007338 0.001085 0.004502 0.000339 0.002210 0.002224 0.001871 0.004021
 1 0.000000 0.050980 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.010030 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000010 0.010540 0.000279 0.000249 0.000191 0.000080 0.000290 0.000664 0.000674 0.000141
 2 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.000010 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
 3 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.000010 0.002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
```

FIG. 13a-8

```
 4 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002428 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.000010 0.002799 0.002968 0.000380 0.002515 0 004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000874 0.000141
 5 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0 002428 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.000010 0 002799 0.002968 0.000380 0.002515 0.004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
 6 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.000010 0 002799 0.002968 0.000380 0.002515 0 004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
 7 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002426 0.004254 0.000183 0.003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.011930 0.002799 0.002968 0.000380 0.002515 0.005970
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000280 0.000280 0.015880 0.000674 0.000141
 8 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0.002741 0.002428 0.004254 0.000183 0 003017
0.003972 0.002456 0.003397 0.005322 0 002703 0.000725 0.005443 0.000010 0.002799 0.002968 0.000380 0 002515 0.004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
 9 0.000000 0.005426 0.002516 0.002857 0.002264 0.007827 0.004722 0 002741 0.002426 0 004254 0 000183 0 003017
0.003972 0.002456 0.003397 0.005322 0.002703 0.000725 0.005443 0.000010 0.002799 0.002968 0.000380 0 002515 0.004602
0.002960 0.000479 0.000010 0.000312 0.000279 0.000249 0.000191 0.000025 0.000097 0.000664 0.000674 0.000141
```

FIG. 13a-9

## Deletion

**B**

|  | sil |
|---|---|
| sil | 0.072822 |
| a | 0.161046 |
| b | 0.263293 |
| c | 0.032190 |
| d | 0.018230 |
| e | 0.224930 |
| f | 0.004360 |
| g | 0.287205 |
| h | 0.042840 |
| i | 0.174973 |
| j | 0.000070 |
| k | 0.012310 |
| l | 0.189107 |
| m | 0.172646 |
| n | 0.080780 |
| o | 0.177360 |
| p | 0.004340 |
| q | 0.000010 |
| r | 0.182459 |
| s | 0.183513 |
| t | 0.004610 |
| u | 0.191691 |
| v | 0.002010 |
| w | 0.001350 |
| x | 0.156087 |
| y | 0.237213 |
| z | 0.000010 |
| 0 | 0.000010 |
| 1 | 0.000010 |
| 2 | 0.000010 |
| 3 | 0.000010 |
| 4 | 0.000010 |
| 5 | 0.000010 |
| 6 | 0.000010 |
| 7 | 0.000010 |
| 8 | 0.000010 |
| 9 | 0.000010 |

**O**

|  | sil |
|---|---|
| sil | 0.086846 |
| a | 0.194318 |
| b | 0.146337 |
| c | 0.156959 |
| d | 0.159377 |
| e | 0.130249 |
| f | 0.148838 |
| g | 0.118878 |
| h | 0.147839 |
| i | 0.161874 |
| j | 0.114188 |
| k | 0.167875 |
| l | 0.125652 |
| m | 0.128615 |
| n | 0.169739 |
| o | 0.182494 |
| p | 0.147606 |
| q | 0.000010 |
| r | 0.112488 |
| s | 0.132361 |
| t | 0.157227 |
| u | 0.152357 |
| v | 0.123944 |
| w | 0.109592 |
| x | 0.157167 |
| y | 0.152881 |
| z | 0.125876 |
| 0 | 0.000010 |
| 1 | 0.000010 |
| 2 | 0.000010 |
| 3 | 0.000010 |
| 4 | 0.000010 |
| 5 | 0.000010 |
| 6 | 0.000010 |
| 7 | 0.000010 |
| 8 | 0.000010 |
| 9 | 0.000010 |

**S**

|  | sil |
|---|---|
| sil | 0.080343 |
| a | 0.147501 |
| b | 0.095018 |
| c | 0.108293 |
| d | 0.103257 |
| e | 0.113145 |
| f | 0.126491 |
| g | 0.114896 |
| h | 0.114976 |
| i | 0.147679 |
| j | 0.000010 |
| k | 0.116405 |
| l | 0.113312 |
| m | 0.107330 |
| n | 0.107703 |
| o | 0.147365 |
| p | 0.104548 |
| q | 0.000220 |
| r | 0.133977 |
| s | 0.157915 |
| t | 0.097456 |
| u | 0.130083 |
| v | 0.103158 |
| w | 0.124114 |
| x | 0.000010 |
| y | 0.104641 |
| z | 0.000890 |
| 0 | 0.144084 |
| 1 | 0.055210 |
| 2 | 0.000010 |
| 3 | 0.000010 |
| 4 | 0.000010 |
| 5 | 0.000010 |
| 6 | 0.000010 |
| 7 | 0.018730 |
| 8 | 0.000010 |
| 9 | 0.000010 |

FIG. 13b

n=1 Context=Silence
Recognized=B

| | | | | |
|---|---|---|---|---|
| Pe: | → B | .655987 | $^{P}B$ | 0.655987 |
| | → D | 04988 | $^{P}D$ | 0.011988 |
| | → E | 011813 | $^{P}E$ | 0.011813 |
| In: | F | 0.010596 | $^{I}E$ | 0.010596 |
| De: | | 0.072822 | $^{D}$ | 0 072822 |

$^{P}B$ 0.655987
n=2 Context=B
Recognized=O

| | | | | |
|---|---|---|---|---|
| Pe: | → B | 0.023924 | $^{P}BB$ | 0.015694 |
| | → O | 0.618126 | $^{P}BO$ | 0.405482 |
| | → W | 0.016075 | $^{P}BW$ | 0.010545 |
| In: | → B | 0.083845 | $^{I}BB$ | 0.055001 |
| | → O | 0.076405 | $^{I}BO$ | 0.050121 |
| De: | | 0.146337 | $^{D}B$ | 0.095995 |

$^{P}BO$ 0.405482
n=3 Context=0
Recognized=S

| | | | | |
|---|---|---|---|---|
| Pe: | → A | 0.011723 | $^{P}BOA$ | 0.004753 |
| | → O | 0.041892 | $^{P}BOC$ | 0.016986 |
| | → S | 0.539338 | $^{P}BOS$ | 0.218691 |
| In: | → O | 0.160171 | $^{I}BOO$ | 0.064946 |
| | → S | 0.042027 | $^{I}BOS$ | 0.017041 |
| De: | | 0.147365 | $^{D}BO$ | 0.059754 |

FIG. 14

$^{D}B$ 0.095995
n=3 Context=B
Recognized=S

| | | | | | |
|---|---|---|---|---|---|
| Pe: | → B | 0.023924 | $^{P}BB$ | 0.001505 | |
| | → S | 0.618126 | $^{P}BS$ | 0.062869 | invalid |
| De: | | 0.016075 | $^{P}B$ | 0.009121 | |

$^{D}$ 0.072822
n=2 Context=Silence
Recognized=O

| | | | | |
|---|---|---|---|---|
| Pe: | → O | 0.873477 | $^{P}O$ | 0.063608 |
| De: | | 0.086846 | $^{D}$ | 0.006324 |

FIG. 14a

$^{P}$O 0.63608

n=3 Context=O

Recognized=S

| | | | | |
|---|---|---|---|---|
| Pe: | → A | 0.011723 | $^{P}$CA | 000746 |
| | → O | 0.041892 | $^{P}$OO | 002665 |
| | → S | 0.539338 | $^{P}$OS | .034306 |
| In: | → O | 0.160171 | $^{I}$OO | .010188 |
| | S | 0.042027 | $^{I}$OS | .002673 |
| De: | | 0.147365 | $^{D}$O | .009374 |

$^{I}$BB 0.055001

n=2 Context=B

Recognized=O

| | | | | | |
|---|---|---|---|---|---|
| Pe: | → B | 0.023924 | $^{P}$BBB | | invalid |
| | → O | 0.618126 | $^{P}$BBO | 0.033998 | " |
| | → W | 0.016075 | $^{P}$BBW | | " |
| In. | B | 0.083845 | $^{I}$BBB | 0.004612 | " |
| | O | 0.076405 | $^{I}$BBO | 0.004202 | " |

$^{P}$BO 0.058121

n=2 Context=0

Recognized=O

| | | | | |
|---|---|---|---|---|
| Pe: | O | 0.577351 | $^{P}$BOO | 0.028937 |
| In: | O | 0.173974 | $^{I}$BOO | 0.008720 |

FIG. 14b

$^{P}$BOO 0.028937
n=3 Context=0
Recognized=S

| | | | | |
|---|---|---|---|---|
| Pe: | → A | 0.011723 | $^{P}$BOOA | 0.000339 |
| | → O | 0.041892 | $^{P}$BOOO | 0.001212 |
| | → S | 0.539338 | $^{P}$BOOS | 0.015607 |
| In: | → O | 0.160171 | $^{I}$BOOO | 0.004635 |
| | → S | 0.042027 | $^{I}$BOOS | 0.001216 |
| De: | | 0.147365 | $^{D}$BOO | 0.004264 |

$^{P}$BB 0.015694
n=3 Context=0
Recognized=S

| | | | | |
|---|---|---|---|---|
| Pe: | A | 0.011723 | $^{P}$BBA | invalid |
| | O | 0.041892 | $^{P}$BDA | " |
| | S | 0.539338 | $^{P}$BBS | " |
| In: | O | 0.160171 | $^{I}$BBO | " |
| | S | 0.042027 | $^{I}$BBS | " |
| De: | | 0.147365 | $^{D}$BBB | 0.002313 |

FIG. 14c

$^{P}$D 0.011988

n=2 Context=D

Recognized=O

Pe: → D 0.017020 $^{P}$DD 0.000204

→ E 0.027756 $^{P}$DE 0.000333

→ O 0.504343 $^{P}$DO 0.006046

In: → P 0.017458 $^{P}$DP 0.000209

→ Y 0.012972 $^{P}$DY 0.000156

→ D 0.060609 $^{I}$DD 0.000727

→ E 0.021489 $^{I}$DE 0.000258

→ O 0.051673 $^{I}$DO 0.000619

→ P 0.016058 $^{I}$DP 0.000193

De: 0.159377 $^{D}$D 0.001911

FIG. 14d

$^{P}$E 0.011873

n=2 Context=F

Recognized=O

| | | | | |
|---|---|---|---|---|
| Pe: | E | 0.036376 | $^{P}$FF | |
| | I | 0.011255 | $^{P}$EI | |
| | N | 0.014544 | $^{P}$EN | |
| | O | 0.513147 | $^{P}$EO | 0.006062 |
| | P | 0.024011 | $^{P}$EP | |
| | T | 0.010950 | $^{P}$ET | |
| In: | E | 0.127255 | $^{I}$FF | |
| | O | 0.054643 | $^{I}$EO | |
| | P | 0.018788 | $^{I}$EP | |
| De: | | 0.130249 | $^{D}$E | 0.001539 |

$^{I}$E 0.010596

n=1 Context=E

Recognized=B

| | | | | |
|---|---|---|---|---|
| Pe: | A | 0.011145 | $^{P}$EA | |
| | B | 0.200916 | $^{P}$EB | 0.002129 |
| | D | 0.017673 | $^{P}$ED | |
| | E | 0.097556 | $^{P}$EE | |
| | P | 0.048040 | $^{P}$EP | |
| | S | 0.011736 | $^{P}$ES | |
| | T | 0.022734 | $^{P}$ET | |
| In: | B | 0.011675 | $^{I}$EB | |
| | D | 0.014182 | $^{I}$ED | |
| | E | 0.217112 | $^{I}$EE | |
| | P | 0.036343 | $^{I}$EP | |
| | T | 0.018599 | $^{I}$ET | |
| De: | | 0.224930 | $^{D}$E | |

FIG. 14e

```
D B
I BOO
D

P EO

P DO   0.0060461
n=3    Context=0
Recognized=S

Pe:  → A  0.011723        P DOA  0.000071
     → O  0.041892        P DOO  0.000253
     → S  0.539338        P DOS  0.003261
In:  → O  0.160171        I DOO  0.000968
     → S  0.042027        I DOS  0.000254
De:       0.147365        D DO   0.000891
```

FIG. 14f

| operation | prefix | confusion value |
|---|---|---|
| P | BOS | 0.218691 |
| I | BOO | 0.064946 |
| D | BO | 0.059754 |
| P | OS | 0.034306 |
| P | BOSS | 0.015607 |
| I | OO | 0.010188 |
| D | O | 0.009374 |
| P | BOA | 0.004753 |
| I | BOOO | 0.004635 |
| P | DOS | 0.003261 |
| I | OS | 0.002673 |
| P | OO | 0.002665 |
| D | BBB | 0.002323 |

FIG. 15

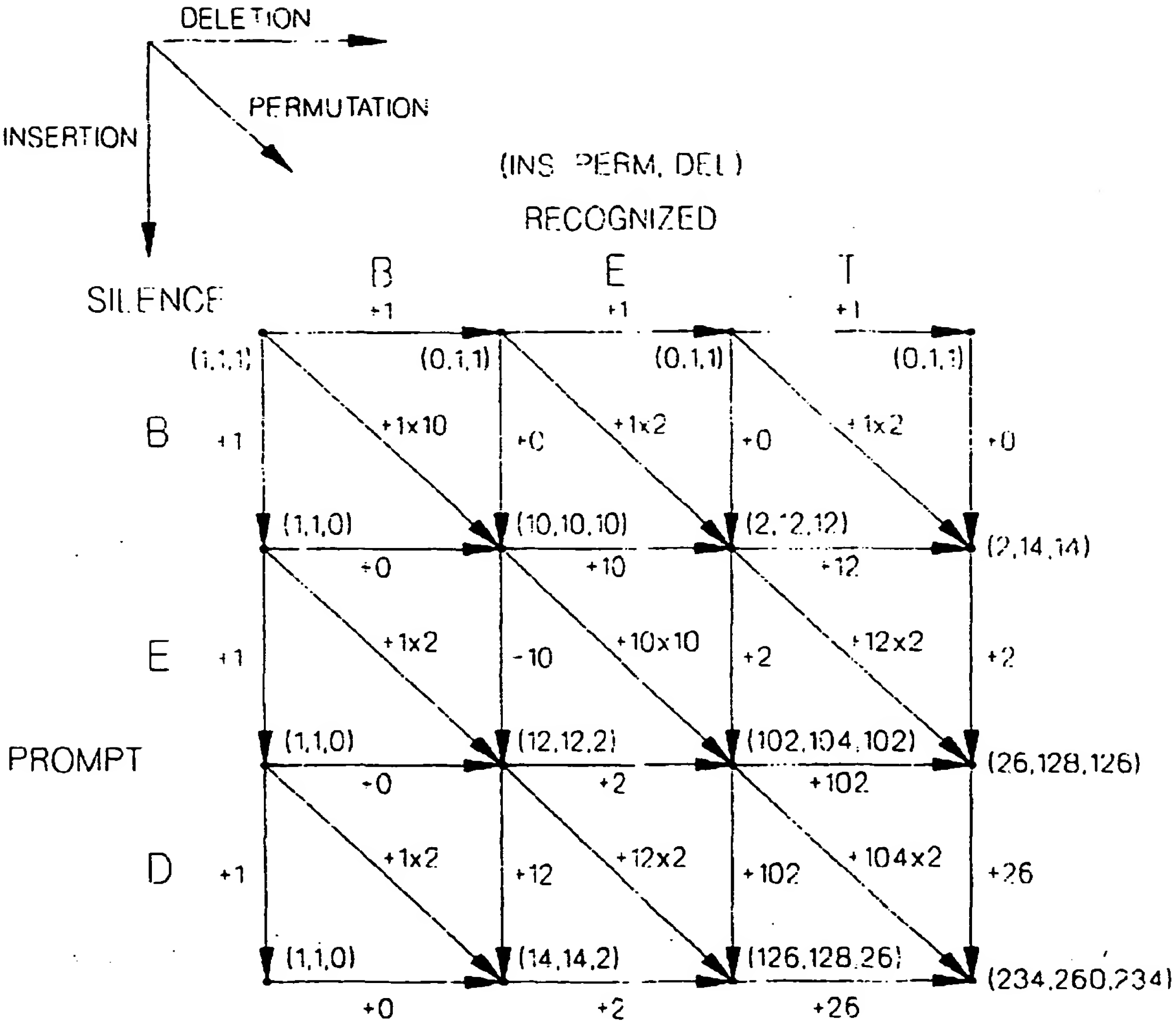

DELETION
PERMUTATION
INSERTION
(INS. PERM. DEL)
RECOGNIZED
B
E
T
SILENCE
+1
+1
+1
(1,1,1)
(0,1,1)
(0,1,1)
(0,1,1)
B
+1
+1x10
+0
+1x2
+0
+1x2
+0
(1,1,0)
(10,10,10)
(2,12,12)
(2,14,14)
+0
+10
+12
E
+1
+1x2
-10
+10x10
+2
+12x2
+2
(1,1,0)
(12,12,2)
(102,104,102)
(26,128,126)
PROMPT
+0
+2
+102
D
+1
+1x2
+12
+12x2
+102
+104x2
+26
(1,1,0)
(14,14,2)
(126,128,26)
(234,260,234)
+0
+2
+26

FIG. 16

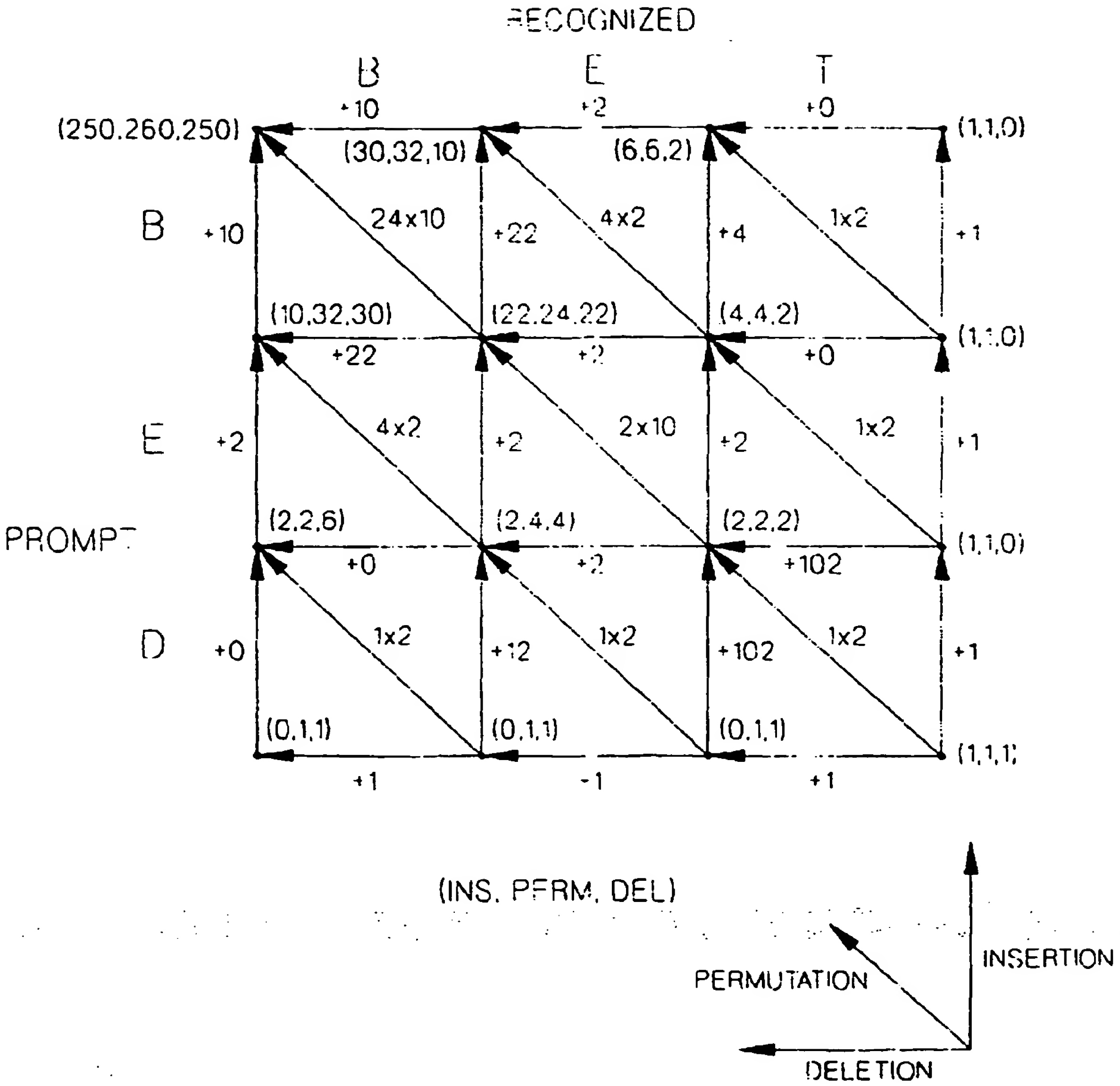
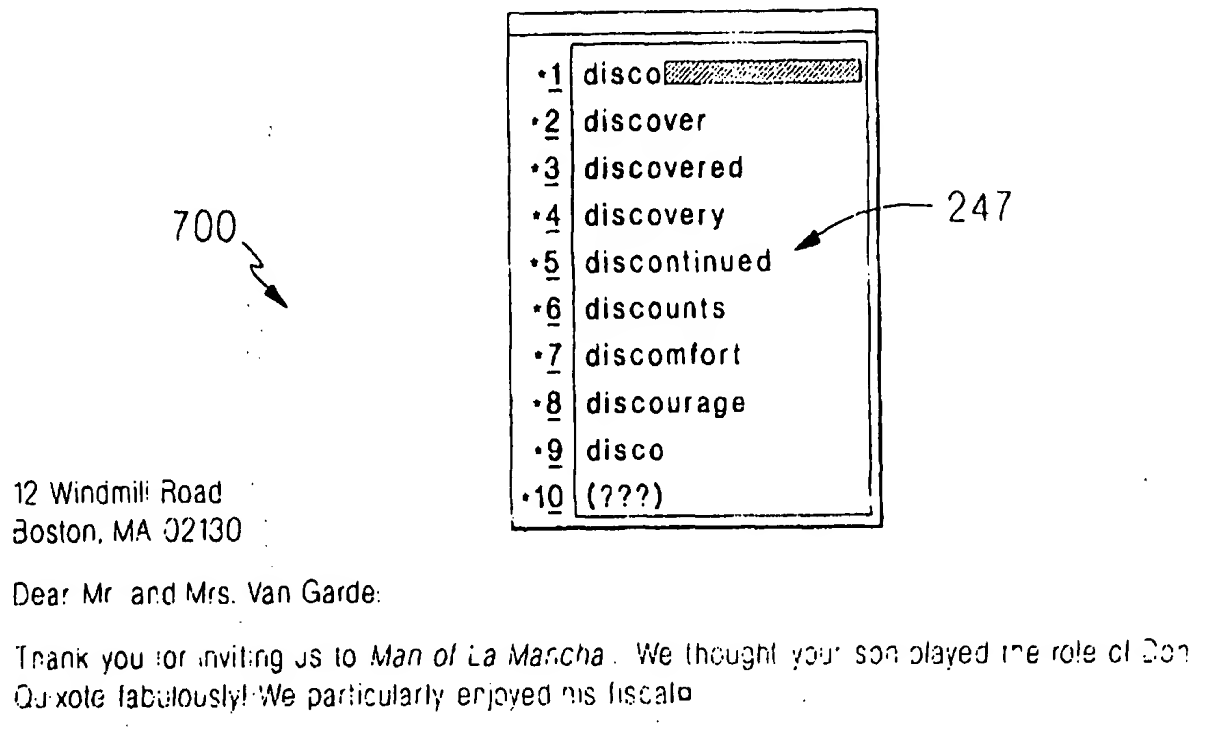

RECOGNIZED
B
E
T
+10
+2
+0
(250,260,250)
(30,32,10)
(6,6,2)
(1,1,0)
B
+10
24x10
+22
4x2
+4
1x2
+1
(10,32,30)
(22,24,22)
(4,4,2)
(1,1,0)
+22
+2
+0
E
+2
4x2
+2
2x10
+2
1x2
+1
PROMPT
(2,2,6)
(2,4,4)
(2,2,2)
(1,1,0)
+0
+2
+102
D
+0
1x2
+12
1x2
+102
1x2
+1
(0,1,1)
(0,1,1)
(0,1,1)
(1,1,1)
+1
-1
+1
(INS, PERM, DEL)
PERMUTATION
INSERTION
DELETION

FIG. 16a



- •1 disco
- •2 discover
- •3 discovered
- •4 discovery
- •5 discontinued
- •6 discounts
- •7 discomfort
- •8 discourage
- •9 disco
- •10 (???)

12 Windmill Road
Boston, MA 02130

Dear Mr and Mrs. Van Garde:

Thank you for inviting us to *Man of La Mancha*. We thought your son played the role of Don Quixote fabulously! We particularly enjoyed his fiscalo

FIG. 17

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 762 385 A3

# (12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
09.09.1998 Bulletin 1998/37

(43) Date of publication A2:
12.03.1997 Bulletin 1997/11

(21) Application number: 96306255.9

(22) Date of filing: 29.08.1996

(51) Int Cl.$^6$: **G10L 5/06**

(84) Designated Contracting States:
DE FR GB IT

(30) Priority: 30.08.1995 US 521543
13.11.1995 US 559190

(71) Applicant: **Dragon Systems Inc.**
Newton, Massachusetts 02160 (US)

(72) Inventors:
- **Gadbois, Gregory J.**
  Newton, Massachusetts 02160 (US)
- **Van Even, Stijn A.**
  Newton, Massachusetts 02160 (US)

(74) Representative: **Deans, Michael John Percy**
**Lloyd Wise, Tregear & Co.,**
**Commonwealth House,**
**1-19 New Oxford Street**
**London WC1A 1LW (GB)**

## (54) Speech recognition

(57) A method of speech recognition includes recognizing a first utterance, recognizing a second utterance having information that is related to the first utterance, and determining the most probable first and second utterances based on stored information about valid relationships between possible first and second utterances. The recognized first utterance may be recognized continuously and the recognized second utterance may be recognized discretely. The determination of the most probable utterances may include creating a list of possible utterances that could be confused with a recognized utterance and rerecognition of a list of possible utterances against an utterance

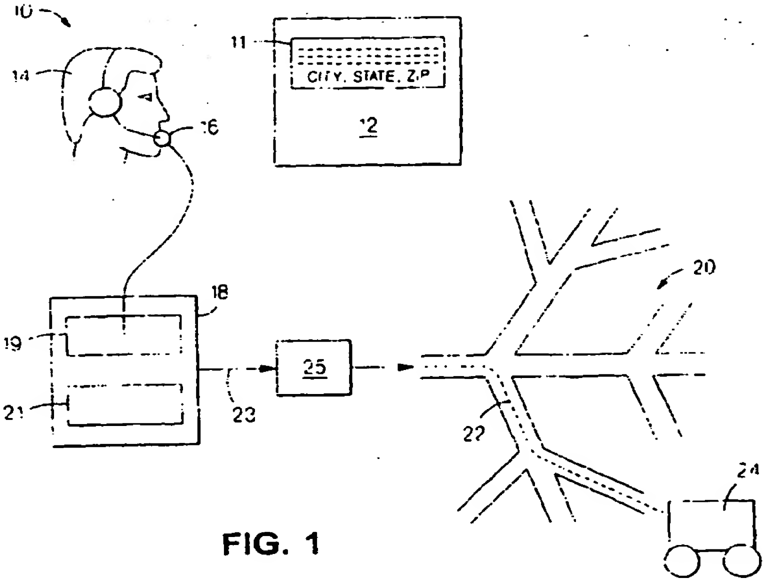


FIG. 1

EP 0 762 385 A3

Printed by Jouve, 75001 PARIS (FR)

European Patent Office

## EUROPEAN SEARCH REPORT

Application Number
EP 96 30 6255

### DOCUMENTS CONSIDERED TO BE RELEVANT

| Category | Citation of document with indication, where appropriate, of relevant passages | Relevant to claim | CLASSIFICATION OF THE APPLICATION (Int.Cl.6) |
|---|---|---|---|
| X,P | WO 96 13030 A (BRITISH TELECOMM ;ATTWATER DAVID JOHN (GB); WHITTAKER STEVEN JOHN)<br>* page 6, line 33 - page 11, line 11; claim 20; figures 1,2 *<br>--- | 1,31 | G10L5/06 |
| X | OCCENA L G ET AL: "A LOGIC-BASED FRAMEWORK FOR ADDRESS INTERPRETATION AND RECTIFICATION"<br>COMPUTERS IN INDUSTRY,<br>vol. 20, no. 1, 1 July 1992,<br>pages 63-73, XP000288646<br>* paragraph 3.1 *<br>* page 69, right-hand column, line 4 - line 35 *<br>--- | 31 | |
| A | EP 0 645 757 A (XEROX CORP)<br>* page 7, line 49 - page 9, line 4 *<br>--- | 1 | |
| A | EP 0 655 732 A (AT & T CORP)<br>* claim 1 *<br>--- | 1 | TECHNICAL FIELDS SEARCHED (Int.Cl.6) |
| A | TAKAYUKI YAMAOKA ET AL: "METHOD FOR PREDICTING UTTERANCES USING A LAYERED PLAN RECOGNITION MODEL-DISAMBIGUATION OF SPEECH RECOGNITION CANDIDATES BY FOCUSING ON SPEAKER INTENTIONS"<br>SYSTEMS & COMPUTERS IN JAPAN,<br>vol. 25, no. 4, April 1994,<br>pages 74-91, XP000476966<br>* paragraph 4.1; figures 4,5 *<br>----- | 1 | G10L<br>G06K |

~~The present search report has been drawn up for all claims~~

| Place of search | Date of completion of the search | Examiner |
|---|---|---|
| THE HAGUE | 3 March 1998 | WANZEELE, R |

CATEGORY OF CITED DOCUMENTS

- X : particularly relevant if taken alone
- Y : particularly relevant if combined with another document of the same category
- A : technological background
- O : non-written disclosure
- P : intermediate document

- T : theory or principle underlying the invention
- E : earlier patent document, but published on, or after the filing date
- D : document cited in the application
- L : document cited for other reasons
- & : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

European Patent
Office

Application Number
EP 96 30 6255

## CLAIMS INCURRING FEES

The present European patent application comprised at the time of filing more than ten claims.

☐ Only part of the claims have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims and for those claims for which claims fees have been paid, namely claim(s):

☐ No claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims.

## LACK OF UNITY OF INVENTION

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

see sheet B

☐ All further search fees have been paid within the fixed time limit. The present European search report has been drawn up for all claims.

☐ Only part of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the inventions in respect of which search fees have been paid, namely claims:

☒ None of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the invention first mentioned in the claims, namely claims:

1-28, 31

European Patent
Office

## LACK OF UNITY OF INVENTION
## SHEET B

Application Number
EP 96 30 6255

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

1. Claims: 1-28,31

   Recognizing a first utterance and a second utterance,related to the first utterance; recognizing ambiguous inputs.

2. Claims: 29,30

   Generating a choice list from a continuously recognized utterance.

3. Claims: 32,33

   Training a speech recognizer.

4. Claims: 34-36

   Displaying word choices during speech recognition.

**This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record**

# BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** ______________________________

**IMAGES ARE BEST AVAILABLE COPY.**
**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**